

SECURITY SYSTEM OF A RELATIONAL DATA BASE SYSTEM

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

By
SATISH K. GOEL

to the
COMPUTER SCIENCE PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
JULY, 1978

DEFENSE
CENTRAL LIBRARY

Acc. No.

54948.

22 AUG 1978

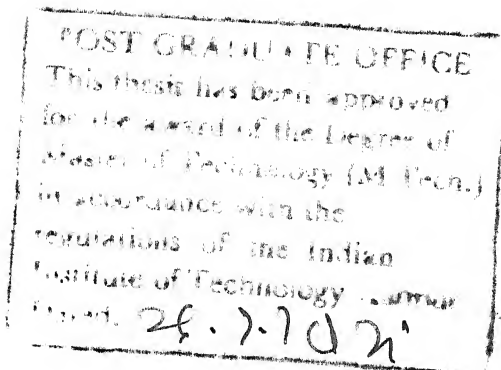
CSP-1970-M-GOE-SEC

CERTIFICATE

CERTIFIED that the work entitled SECURITY SYSTEM OF A RELATIONAL
DATA BASE SYSTEM has been carried out under my supervision by Sri S.
K. Goel and it has not been submitted elsewhere for a degree.

Kanpur
July 19, 1978

R. Sankar
R. Sankar
Professor of Computer Sciences
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR



ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to all my teachers who introduced me to the area of computers and specially to my thesis supervisor, Professor R. Sankar, whose continued guidance and inspiration helped me throughout the execution of this project.

Thanks are also due to my friends Deepak Ghanekar, Sat Pal and S. Agarwala who have worked on this project and with whom I had lively discussions about the progress of the project.

Credit for expert typing goes to Mr. H.K. Nathani, his patience is gratefully acknowledged.

Kanpur
July 19, 1978

- Satish K. Goel

LIST OF CONTENTS

Chapter 1	INTRODUCTION	1
2	PLANNING OF THE SECURITY SYSTEM	5
3	INITIALISING THE SECURITY TABLES	28
4	MODIFYING THE SECURITY TABLES	34
5	PRINTING THE SECURITY TABLES	47
6	REMOVING A RELATION	54
	REFERENCES	61
	Appendix 1	62
	Appendix 2	64
	Appendix 3	67

ABSTRACT

This thesis deals with the design and implementation of a security mechanism for a relational data base. It also deals with removing a relation from the data base and reorganization of the data base.

It is a part of the project to implement a relational data base system on TDC-316. This thesis alongwith (Ref. 5) completes the implementation of functions falling in the domain of the data base administrator, i.e., controlling the overall view of the data, reorganization of the data base and ensuring security of the data base.

1. INTRODUCTION

One of the most important aspects of a data base system is to provide for adequate security of the data base. Since a data base generally contains the lump sum of data useful for any organization and various users keep interacting with the same, retrieving and updating the data placed in a central place, it is highly important that there be a mechanism for stopping illegal use of the data. Hence appropriate checks must be placed on every user of the data base. Moreover, there must be ways of checking out users who tend to use the system in an unauthorised manner and also indicate the same whenever asked to do so, so that defaulting users may be asked for proper explanations.

The description of a relational data base system and the various terms related to it as well as hardware and software tools available are very well described in a separate thesis (Reference 5). A data base as we know, is the collection of data useful for many applications and placed in a central place. The various applications extract from the data base their own views of data and application program are independent of the way data is organised in a data base. It is the responsibility of the data base management system to take care of any changes in the physical organization of the data base, to make the application programs immune to any changes taking place in the physical data base organization.

In a relational data base system, all the data is stored in terms of relations, i.e., flat files. A flat file is many similar occurrences of a record which consists of a number of fields.

The security system for a data base can be of varied complexity. The following discussion seeks to illustrate this point: Suppose that our data base contains PERSON-DETAILS as one of the relations as shown in Figure 1-1. A few security implementations (of increasing complexity

Personnel No. P #	Name N	Rank R	Pay P	Confidential Report CR

Figure 1-1.

could be as follows:

1. The user has access to all records of the relation if he is an authorised user, otherwise he has no access to the relation, i.e., either a user has authority to use a relation in entirety or not at all.
2. The user may see name, rank and pay of any person but may not see confidential report of any one. Thus the authority here is of field level i.e., either the user has some particular authority of over a particular field/all records or he has no authority over that field for all records.

3. The user may see the name and rank of any person but may see his pay and confidential report only if his rank is higher than that of the person under consideration. Moreover a user may change the name if the record pertains to his own personnel number. Here we see that the user has some authority over a field for a particular record depending upon his qualification in relation to the other field values for that record. Here the users identity must reveal to the system his personnel number, rank etc., to be able to enforce this type of security scheme.

Moreover there can be other access parameters like location of the accessor, the time and day of the access and the maximum frequency with which an accessor is allowed to access the system etc.

This present thesis describes a security scheme of the field level of a relation as in No. 2 above. The five access privileges associated with a field are as below :

1. READ : In this case, the particular field value can be read for all records for processing.
2. OUTPUT : In this case the particular field value can be printed on a listing device for any record.
3. DELETE : In this case and record of a relation may be deleted if the accessor has DELETE right for all its fields.
4. MODIFY : In this case, the particular field value may be modified for any record.
5. INSERT : In this case, any no. of records can be inserted in a relation if the user has INSERT right over all its fields.

The access in the present system does not depend upon any parameters like location, time and day, frequency etc., because of hardware limitations (because the system has only one access location and there is no timer in the system).

In the last chapter, programs are described for removing a relation from the data base and a discussion on how to reorganise the data base.

2. PLANNING OF THE SECURITY SYSTEM

The previous chapter describes how a security system can be of varied complexity. The most primitive security scheme could be to allow or disallow any user from accessing the data base. Once a user is put through, he shall be able to perform any operations on the data in the data base. But such a scheme is not suitable for any practical data base system because a data base essentially contains data for various applications and we shall like any user to have only limited access to it, to perform the functions falling only in his application area. As the complexity of the security mechanism increases, there will be a corresponding increase in its cost in terms of software development effort and more prominently in terms of the fraction of total run time taken by the security mechanism to respond to any query.

The present security mechanism is as described below:

We know that in a Relational Data Base System, each relation consists of some fields, each identified by a field identification number. A field may be common to many relations, but its field identification number is the same for its occurrence in every relation. In the type of security system provided here, we shall build an authority vector or access rights vector for any user from the security codes supplied by him to the system. The i th element of this vector shall give the rights of the user related to the field with identification No. i . Since TDC-316 is a byte addressable

machine with 8 bit byte, it was decided that each element of the authority vector shall be stored in a byte, whose 8 bits shall be associated with the following rights. Figure 2-1 depicts this association.

1. READ
2. OUTPUT
3. DELETE
4. MODIFY
5. INSERT
6.)
7.) UNUSED
8.)

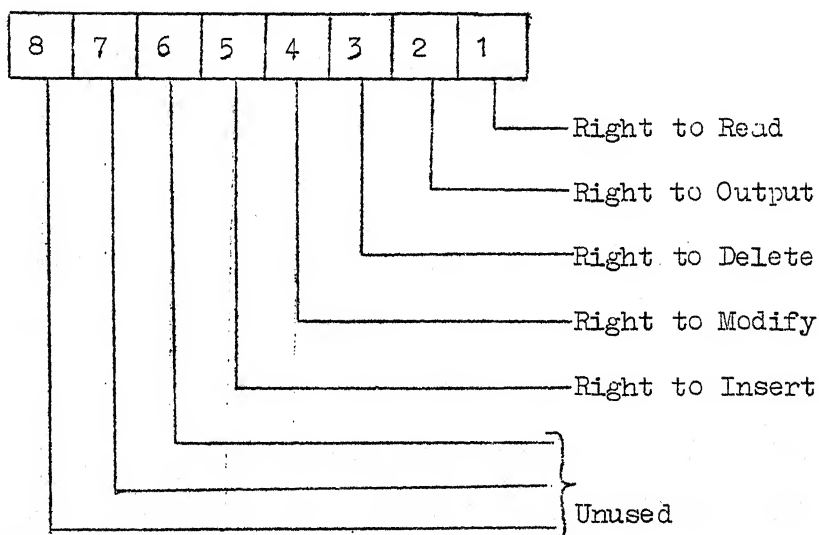


Figure 2-1: Rights associated with byte bits.

If a user has a particular right over a field, the corresponding bit shall be set, else it will be clear. Since the core memory available is very limited and all security codes and authority augmenting vectors cannot be profitably accommodated in core, it was decided to store the security codes and authority augmenting vectors in some area of the disk.

The present disk pack attached to TDC-316 has 203 cylinders. Each cylinder has 10 tracks (0 to 9) and each track is divided into 10 sectors (1 to 10). Each sector has a capacity to store 256 bytes (8 bits each).

Since the upper limit on the number of fields is 256 in the system designed, hence each authority augmenting vector can be accommodated in a sector of the disk.

In our system, we shall have two types of security codes:

1. FIRST SECURITY CODE: This is a code which helps the user to identify himself to the system. This helps the user to get access to the system. This is also called identification number of the user and is unique for each user.
2. AUTHORITY AUGMENTING SECURITY CODE: This code is used for augmenting the authority of the user. Each "Authority Augmenting Code" has an authority augmenting vector associated with it. A user may give more than one authority augmenting code to the system. The authority vector of a user is the logical OR of all those authority augmenting vectors for which he has given the corresponding authority augmenting security codes as input to the system.

Though authority vectors could be associated directly with "First Security Codes", i.e., user's identification numbers, but having two types of security codes as provided is useful on two counts -

1. Even if an unauthorised user breaks through the system by trying various first security codes till he succeeds in getting the right one (even though this is highly improbable), it is just impossible for him to give the correct authority augmenting security code in the first trial. Once he gives the wrong authority augmenting security code (after having succeeded in breaking the cardon of first security codes), the corresponding "First security code" is deleted from the group of valid first security codes and hence his access is again restricted to the system. It is in a way like a two level protection, where any mistake at the second level puts the user out of the first level.

2. In general, many users (at the same level of management) shall have similar authority vectors. Therefore, we can merge their authority vectors into one and give them same authority augmenting security code whereas they have different first security codes. Since an authority augmenting vector takes much larger storage space (one sector) compared to that taken for storing only security code, we shall save in storage space on disk by merging authority augmenting vectors.

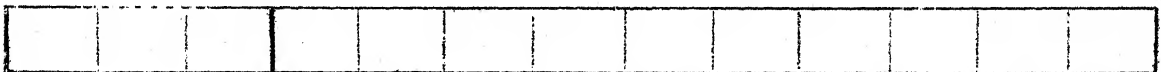
DATA STRUCTURES USED

1. FIRST SECURITY CODES

Since there are many first security codes (one for each user), therefore, to know whether a particular security code belongs to the set of valid codes, either we have to match it with each member of the set to see if it tallies with any one of these, or there should be

some information within the code itself, which indicates its position in the set. So we don't have to search through the entire set for a match but we only need to see whether security code at that particular position matches or not. It was decided that the first three digits of the code shall indicate its position within the array of codes. The length of the rest of the code was chosen to be 10 digits, thus giving a 13 digit security code as shown in Figure 2-2.

The internal storage of each security code is 5 bytes in the format shown in Figure 2-3. The lower four bytes contain the binary equivalent of the number generated from the lower 10 digits of the security code. Bits of the 5th byte contain information telling whether this security code exists or not, whether it was illegally used and if yes in what manner as indicated in Figure 2-3.



First three digits tell the position of the security code in the security codes table.

These lower 10 digits are converted into a number for internal storage taking four bytes.

Figure 2-2. Format of Security Codes.

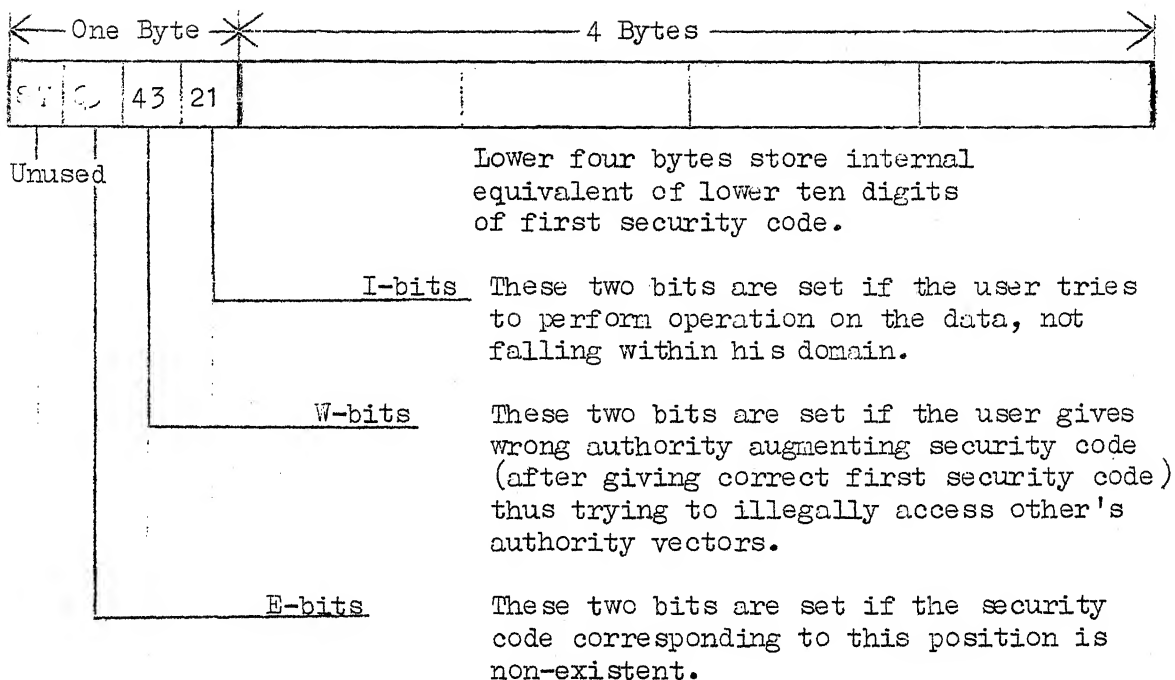


Figure 2-3: Format for internal storage of first security codes.

Since one security code takes five bytes for internal storage, so the number of security codes that can be accommodated in one sector of the disk is $\lfloor 256/5 \rfloor = 51$.

To keep track of the sectors which store first security codes, we have a table shown in Figure 2-4 called "FIRST Security Codes" TABLE DIRECTORY" (FSECDC). As can be seen from the figure, each row of this table takes 14 bytes. Since this table itself is placed in a sector of the disk, so maximum number of rows of this table is $\lfloor 256/14 \rfloor = 18$. Since one row contains information about one sector (which may contain maximum of 51 security codes), so maximum number of First-Security-Codes that can be issued is equal to $18 \times 51 = 918$, which is a fairly large number and sufficient for all practical applications.

The columns of "First Security Codes' Table Directory" have the following meanings:

← 14 Bytes →			
Cylinder No. CYLDRF	Sector & Surface No. SECTRF	10 digits Random No. RDND	
1 2 Bytes	2 Bytes	10 Bytes	The ith row of the table contains relevant information for security codes whose first 3 digit number lie between ix51-50 and (ix51)
2			
3			
4			
:			
:			
15			
16			
17			
18			

Figure 2-4: First Security Codes' Table Directory (FSECDC)

CYLDRF: This column contains the cylinder number of the disk where the sector containing information about the corresponding group of codes is stored. A non-existing entry in this table is indicated by a -ve value of this field for that particular row.

SECTRF: This column contains the surface number and sector number of the sector of the disk containing information about the corresponding group of codes. These two values are packed into one word in the form shown in Figure 2-5, which is the same as that used by the disk controller.

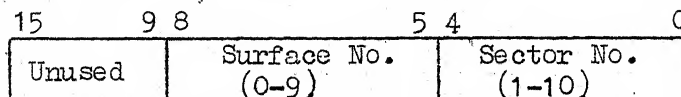


Figure 2-5: Format for putting sector No. & Surface No. in one word.

RDND: This column stores a 10-digit random number for each group of 51 security codes (i.e., one sector). This is used to generate another number from the lower 10 digits of the security code using the routine (WORBLE). The idea is to make the internal number stored on the disk different from the corresponding first security code so that it may be extremely difficult, if not impossible, for any person to compute the first security code of some one else, after he has somehow (though it shall not be possible through normal channels as the disk shall also be protected) read the internally stored security code from the disk. Since this routine WORBLE is changable at the discretion of the DBA, it will be very difficult to know, what transformation this routine is applying on the security code to generate another 10 digit number which is converted into its binary value for internal storage, by the routine GIVSEC.

The only information that is maintained in core, regarding the first security codes, is the address of the sector containing "First Security Codes' Table Directory". There are two variables (one word each) for keeping track of this information:

ISECYL : contains cylinder number of the disk where "First Security Codes' Table Directory" (FSECDC) is stored.

ISECTR : contains sector and surface number of the disk where FSECDC is stored (in the format shown in Figure 2-5).

ISECYL = 196

ISECTR = 35

-	001	003
15	9 8	5 4 0

FSECDG

Sector of disk in
cylinder No. 196,
surface No. 1,
sector No. 3

	CYLDRRF	SECTRF	RDND
1	196	39	1 9 8 6 5 4 3 2 5 7
2	-1	-	-
3	-1	-	-
4	197	01	7 3 2 5 8 6 3 2 1 8
5	-1	-	-
6	-1	-	-
7	-1	-	-
8	-1	-	-
9	-1	-	-
10	-1	-	-
11	197	05	1 3 2 5 4 9 7 8 3 6
12	-1	-	-
13	-1	-	-
14	-1	-	-
15	-1	-	-
16	-1	-	-
17	-1	-	-
18	-1	-	-

← 2 Bytes → 2 Bytes ← 10 Bytes →

Cyl. No. 196
Surface No. 1
Sector No. 7.

Cyl. 197
Surface 0
Sector 1

Cyl. 197
Surface 0
Sector 5

1	48	-
2	:	:
15	48	-
16	0	1 5 9 6 8 2 3 4 5 7
17	48	-
:	:	:
27	48	-
28	03	2 3 0 5 8 2 6 4 1 7
29	0	0 6 8 3 0 0 5 0 2 8
30	48	-
:	:	:
43	48	-
44	0	3 2 5 8 8 6 6 9 7 3
45	48	-
:	:	:
51	48	-

One byte Four bytes

1	48	-
:	:	:
15	48	-
Sec. 16	0	1235407231
code 17	12	0865324321
016... 18	48	-
:	:	:
:	:	:
:	:	:
:	:	:
:	:	:
:	:	:
:	:	:
Sec. code =044	:	:
51	48	-

48	-
:	:
:	:
:	:
:	:
:	:
:	:
48	-
29 0	0783216541
48	-
:	:
:	:
:	:
:	:
:	:
:	:
48	-

Sec.
Code
539..

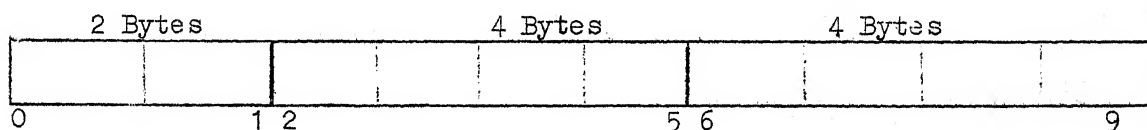
Figure 2-6: (An Instance) of First Security Codes' Tables.

Figure 2-6 shows, how an instance of these tables may look at any moment. This also depicts how the mapping of a security code to its internal stored equivalent is done starting from ISECYL and ISECTR. In the Figure, -1 in the CYLDRF field indicates a non-existing entry. We see that in the figure only two entries exist and contents of these sectors show that only 7-security codes exist, out of which two have been used illegally and hence are unusable.

2. AUGMENTING SECURITY CODES

The augmenting security code also has a length of 13 digits with the first 3 digits giving information regarding the position of a code in the table of authority augmenting security codes.

The internal storage for each security code is 10 bytes as shown in the format in Figure 2-7.



These two bytes contain the cylinder No.

These 4 bytes contain a scrambled no. generated from the 9-bit surface and sector No. using routine VORBLE.

These four bytes contain the internal equivalent of the lower 10 digits of the authority augmenting security code.

Figure 2-7: Format for Internal Storage of Authority Augmenting Security Code.

Again 4 bytes are taken up by the binary equivalent of the internal number generated from the lower 10 digits of the security code, as in the case of first security codes. The remaining six bytes are taken by two fields given below:

Cylinder No. (2 Bytes): We know that there is one authority augmenting vector stored on the disk corresponding to each existing authority augmenting security code. This field gives the cylinder number of the disk which contains the corresponding authority augmenting vector in one of its sectors. A -ve value of this field indicates that the security code corresponding to this position of the table has not been defined by the DBA.

Sector and Surface No. (4-Bytes): This field stores the surface and sector number of the disk (with cylinder number indicated by cylinder number field) where the corresponding authority augmenting vector is stored. Though 2 bytes are enough for this field (in the format shown in Figure 2-5), the reason for providing four bytes is as follows -

As the contents of authority augmenting vectors are of crucial importance, we would like it to be impossible for any intruder to be able to know where his authority augmenting vector is stored and to change the contents of any authority augmenting vector to give him access to the areas prohibited to him. Therefore, instead of storing the surface and sector number of the authority augmenting vector directly, we scramble it to produce another artificial number of 4 bytes from this 9-bit information using a routine (VORBLE) and put its value on the disk. There shall be a routine (DVORBL) to change this four byte value back to the same 9 bits of information. These two routines (VORBLE) and (DVORBL) are changable at the discretion of the DBA but should be such that for each 9-bit input to routine (VORBLE), the 4-byte output produced by it should be such that when fed to the routine (DVORBL), it produces as output the same pattern of 9 bits which was given as input to routine (VORBLE).

One thing to be noted is that there is no information stored regarding wrongful use of any authority augmenting security code. This information is purposeful only for the first security codes so that the users trying to use the system in any illegal manner may be marked and asked for explanations.

With 10 bytes of internal storage being needed for each Authority Augmenting Security Code, the number of codes about which information can be maintained in a sector of the disk is $\lfloor 256/10 \rfloor = 25$. Again to keep track of the sectors which store authority augmenting security codes, we have an AUGMENTING SECURITY CODES' TABLES DIRECTORY (ASECDC) as shown in Figure 2-8. This has exactly the same format as FSECDC shown in Figure 2-4 and the fields have exactly similar explanations. It is also accommodated in a sector of the disk and has maximum of 18 entries. So maximum number of authority augmenting security codes that can be issued is $18 \times 25 = 450$.

Again the only information that is maintained in core, regarding authority augmenting security codes, is the address of the sector containing Augmenting Security Codes' Table Directory. The two variables (one word each) holding this information are as below -

ASECYL: contains the cylinder number of the disk where the Augmenting Security Code's Table Directory (ASECDC) is stored.

ASECTR: contains the surface and sector number of the disk where ASECDC is stored (in the format of Figure 2-5).

14 bytes		
Cylinder No. CYLDRA	Sector & Surface No. SECTRA	10-digit Random No. RNDRA
2 bytes	2 Bytes	10 Bytes
1		
2		
3		
4		
...		
...		
...		
15		
16		
17		
18		

Figure 2-8: Augmenting Security Codes' Table Directory
(ASECDC)

Figure 2-9 shows an instance of authority augmenting security codes tables with only 3 authority augmenting security codes existing.

Since the DBA (data base administrator) has also to approach the system through a "first security code" (no separate provision being made for him), to identify the first security code of the DBA, we have a variable DBSC.1 (one word) which stores the first three digits (converted to equivalent binary form) of DBA's first security code.

ALGORITHM FOR SECURITY CHECKING

When a user indicates his interest in using the data base system, only a small part of the programs get loaded from disk into main memory. This part of the program asks the user to give his security codes and then if found valid, it loads the appropriate programs depending upon whether the user is a Data Base Administrator or some other user. If user is the Data Base Administrator, then the programs loaded are as below:

ASECYL = 200

ASECTR = 03

-	0	3
9	8	5 4 0

ASECDC

Sector of the
Disk in Cylinder
No. 200
Surface No. 0
Sector No. 3

	CYLDRA	SECTRA	RDNDA
1	-1	-	-
2	-1	-	-
3	197	08	5678432109
4	-1	-	-
5	-1	-	-
6	-1	-	-
7	-1	-	-
8	-1	-	-
9	-1	-	-
10	-1	-	-
11	-1	-	-
12	-1	-	-
13	-1	-	-
14	-1	-	-
15	-1	-	-
16	-1	-	-
17	-1	-	-
18	-1	-	-

Cylinder No. 197,
Surface No. 0,
Sector No. 8

	Cylinder No.	Sector & Surface No. (Scrambled value)	Security Code
1	-1	-	-
2	-1	-	-
3	198	-	-
4	-1	Scrambled value of 69(2,5)	1987612345
5	:	:	:
6	:	:	:
7	:	:	:
16	-1	-	-
17	196	Scrambled value of 100(3,4)	1765431829
18	-1	-	-
19	-1	-	-
20	199	Scrambled value of 04(0,4)	1976523498
21	:	:	:
22	:	:	:
23	:	:	:
24	:	:	:
25	-1	-	-

Cylinder No. 198
Surface No. 2
Sector No. 5

Authority
vector for
Sec. code
= 105.....

Cylinder No. 196
Surface No. 3
Sector No. 4

Auth.
Aug.
Vec.
for Sec.
Code =
119.....

Cylinder No. 195
Surface No. 0
Sector No. 4

Auth.
vec.
of sec.
code =
122.....

Figure 2-9: An Instance of Authority Augmenting Security Codes' Tables.

1. Program for deleting any security codes.
1. 2. Programs for changing any security codes or authority vectors.
3. Program for inserting new security codes.
2. 4. Program for printing of the security codes tables.
3. 5. Program for initializing the security codes tables.
4. 6. Program for building of the data base (setting up new relations)
5. 7. Program for removing a relation from the data base.

Otherwise the programs loaded are as below -

1. Program for retrieving any information from the data base.
2. Program for updating information in the data base.
3. Other routines and data used by above two programs.

The flowchart of the program which asks for security codes and loads appropriate programs is shown in Figure 2-10. The following paragraph supplements the information given in the flowchart.

The initial data is the value of ISECYL, ISECTR, ASECYL, ASECTR, DBSC.1, data which tells from where in disk various programs like update, retrieve, security programs, build programs etc. are to be loaded and to which areas of main memory they go. (Since the object program are in absolute machine language form). Only the assembly language equivalent of flowchart (Figure 2-10) is incorporated under KDM with code "DB". This program does rest of the job itself.

The program for retrieval and updating are developed under the separate theses (Reference 1 and 7). The present thesis along with (Ref. 5) where programs for building the data base are described completes the functions carried out by the DBA. Flowchart in Figure 2-11

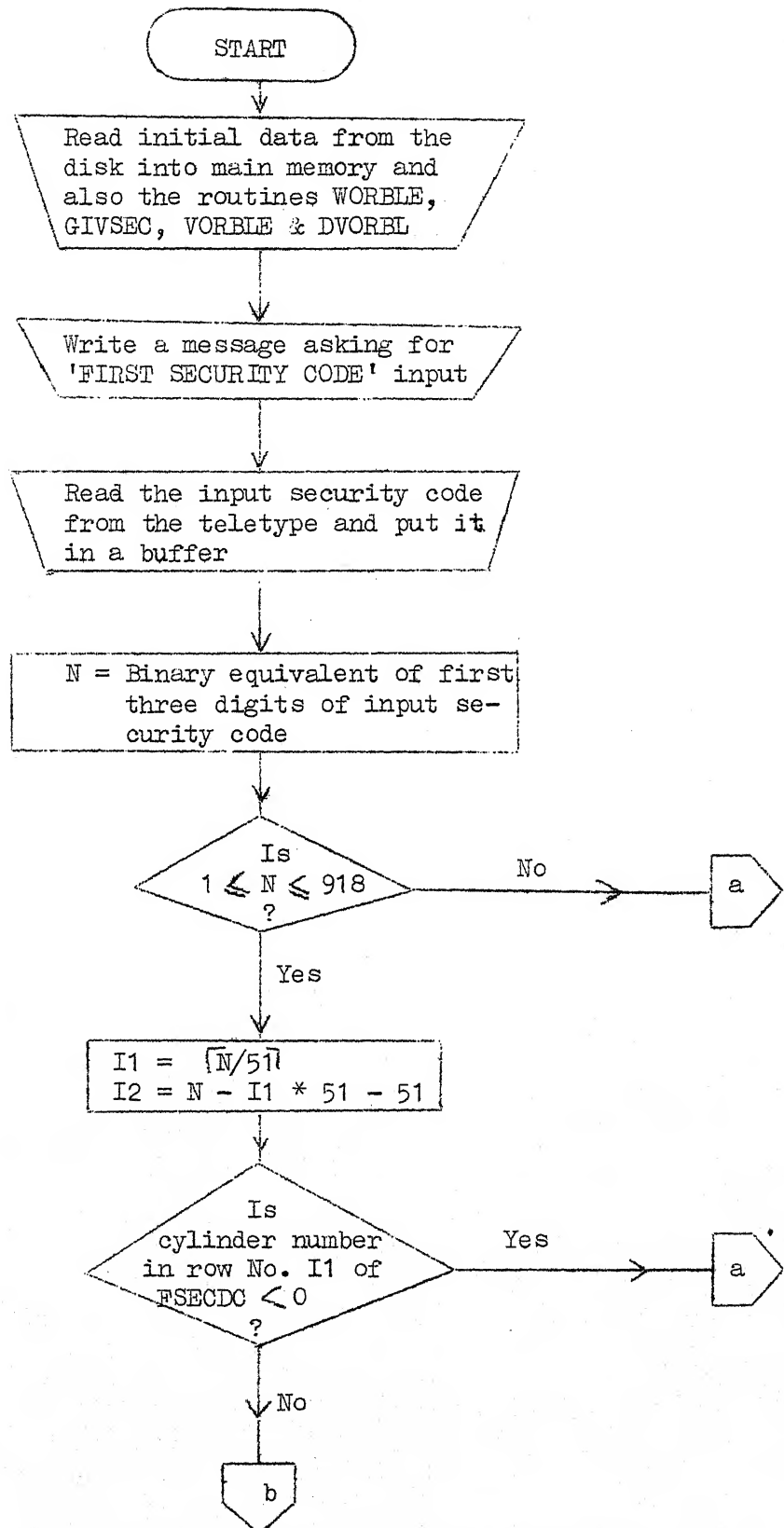


Figure 2-10: Continued on next page.

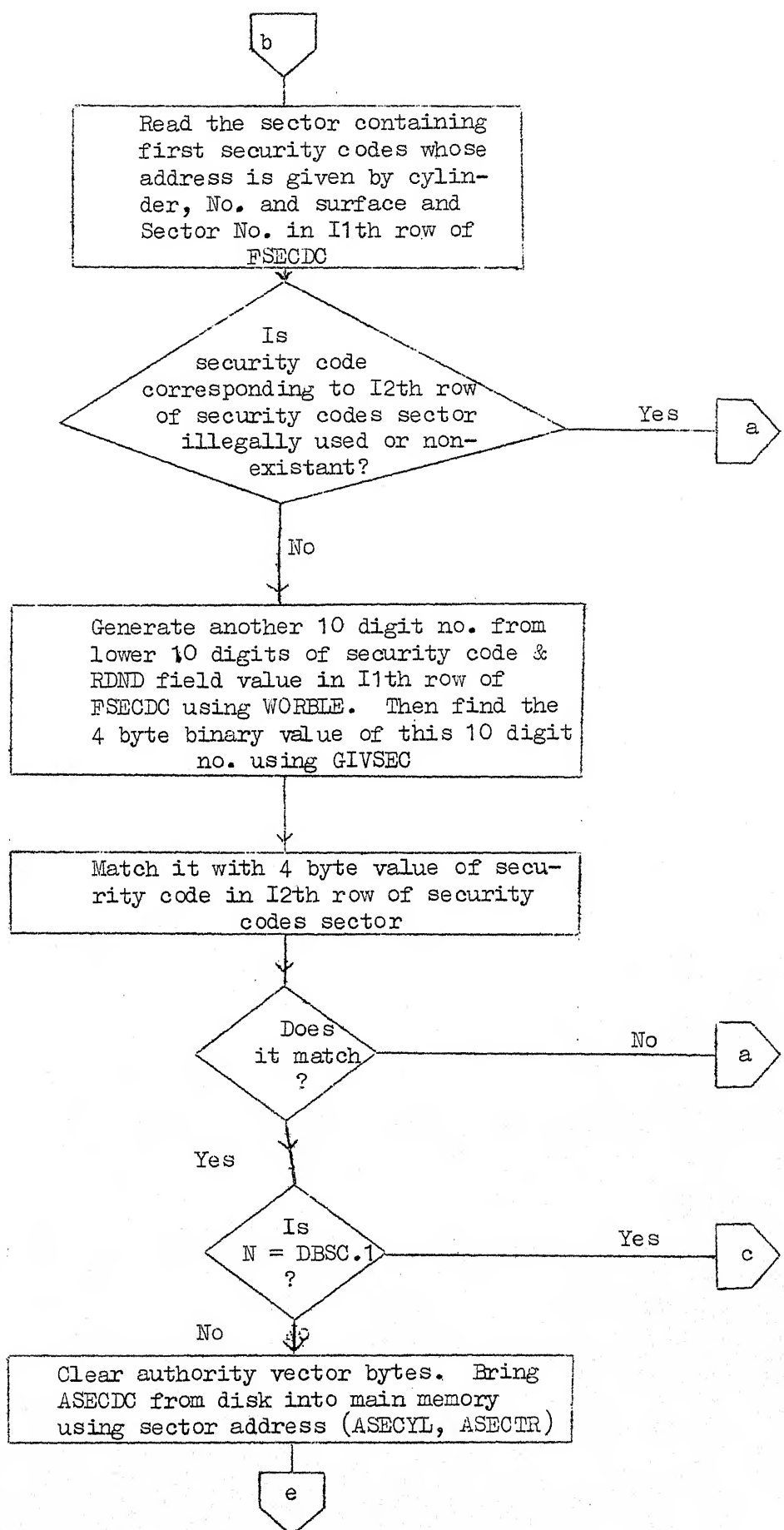


Figure 2-10: Continued on next page.

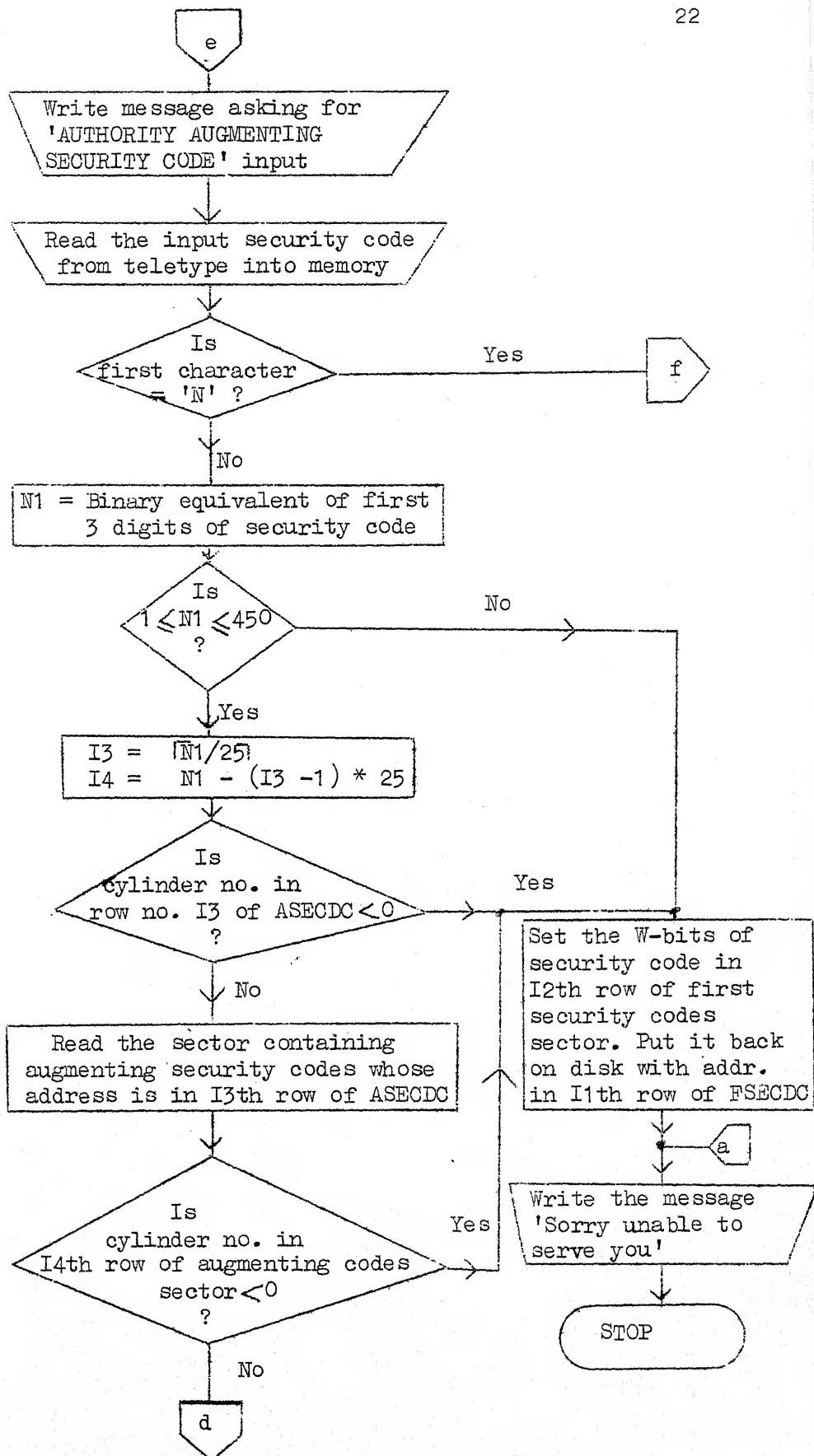


Figure 2-10: Continued on next page.

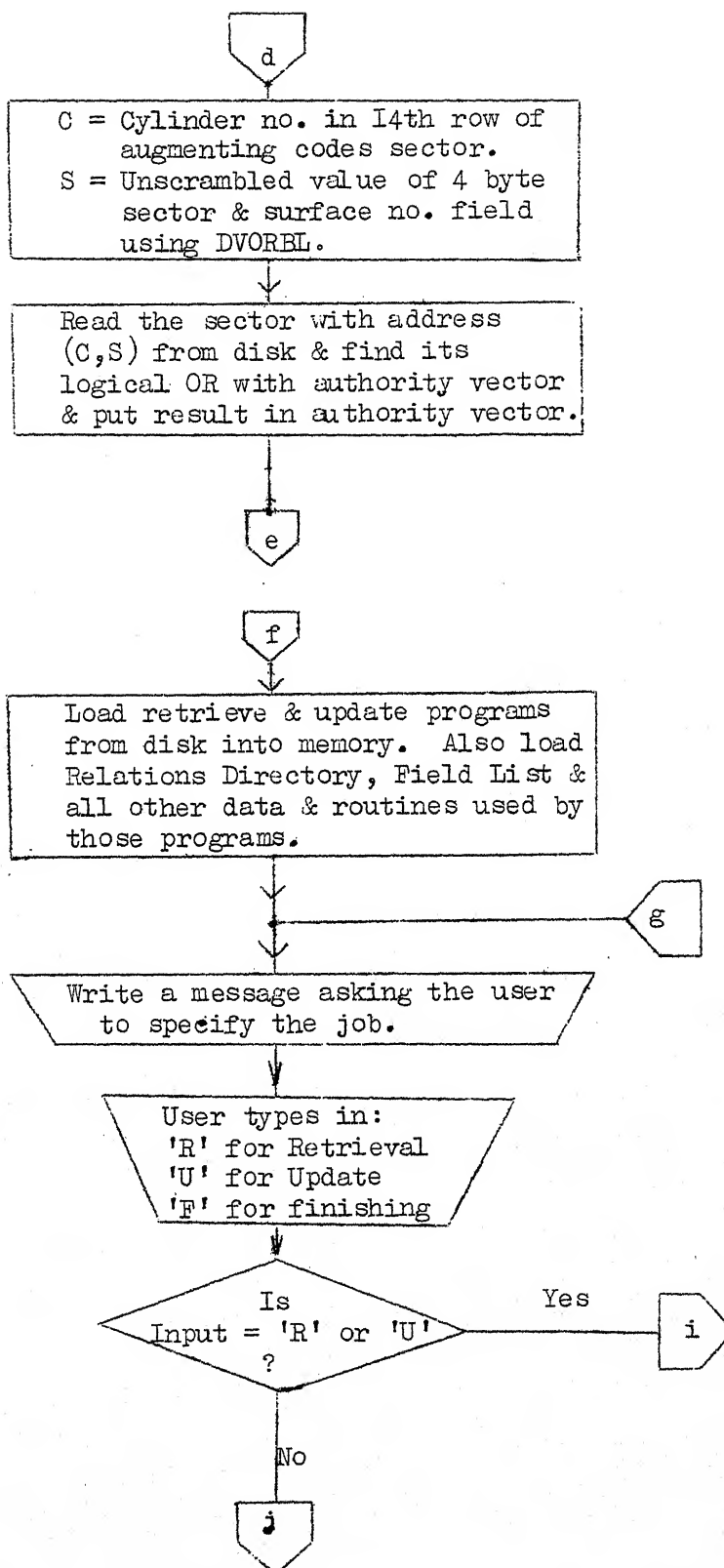


Figure 2-10: Continued on next page.

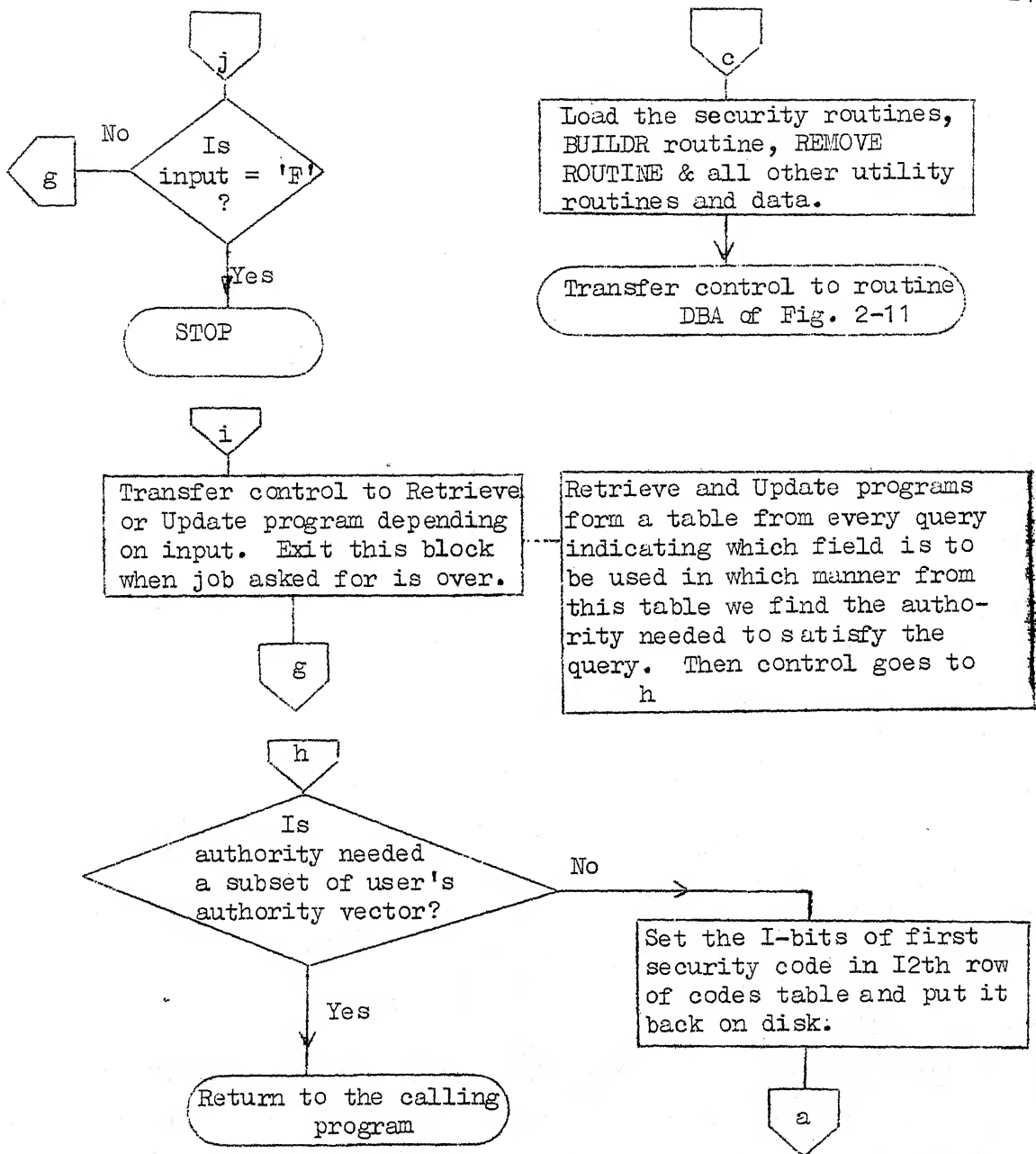


Figure 2-10: Flowchart for checking Security Codes.

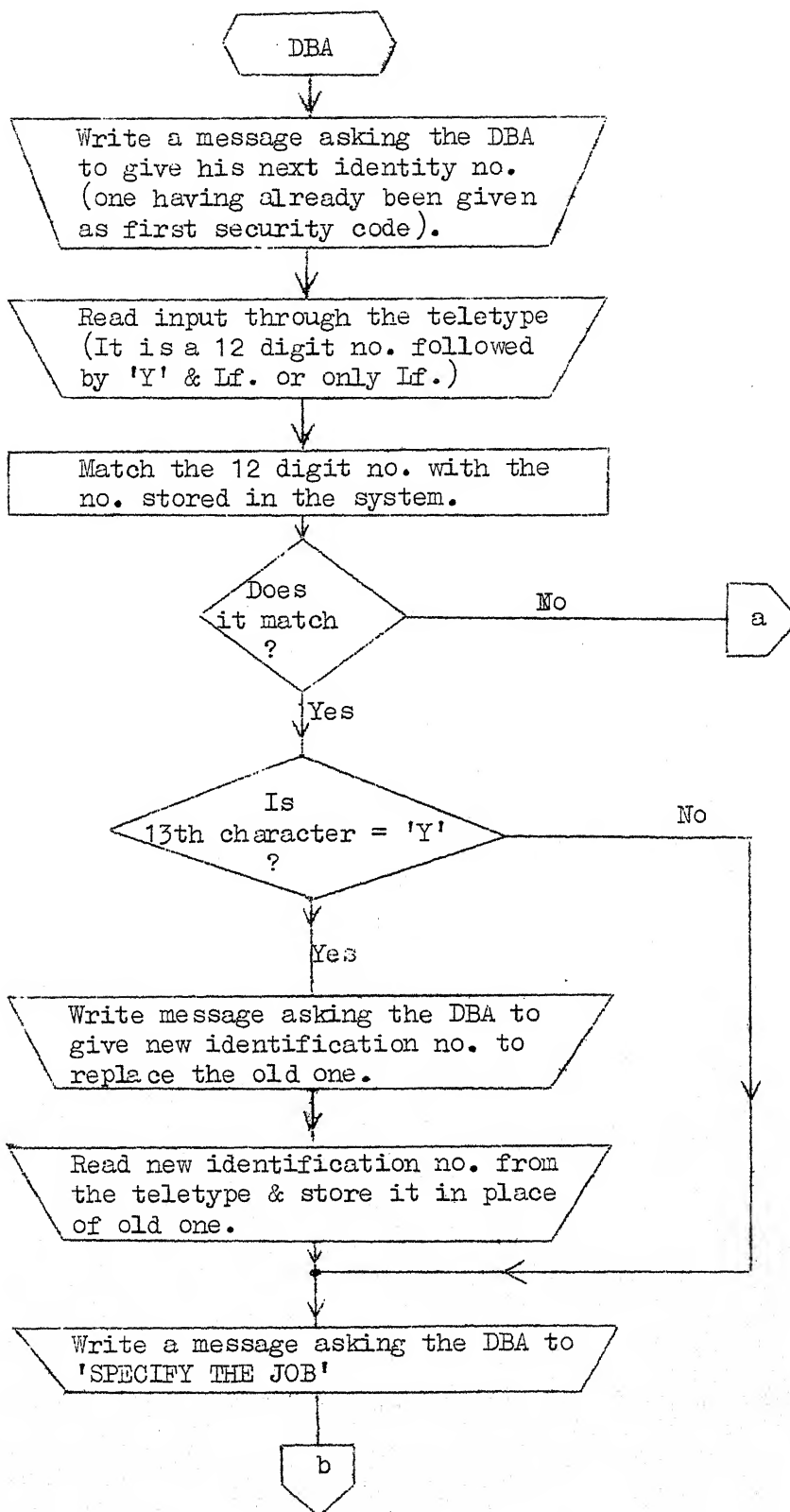


Figure 2-11. Continued on next page.

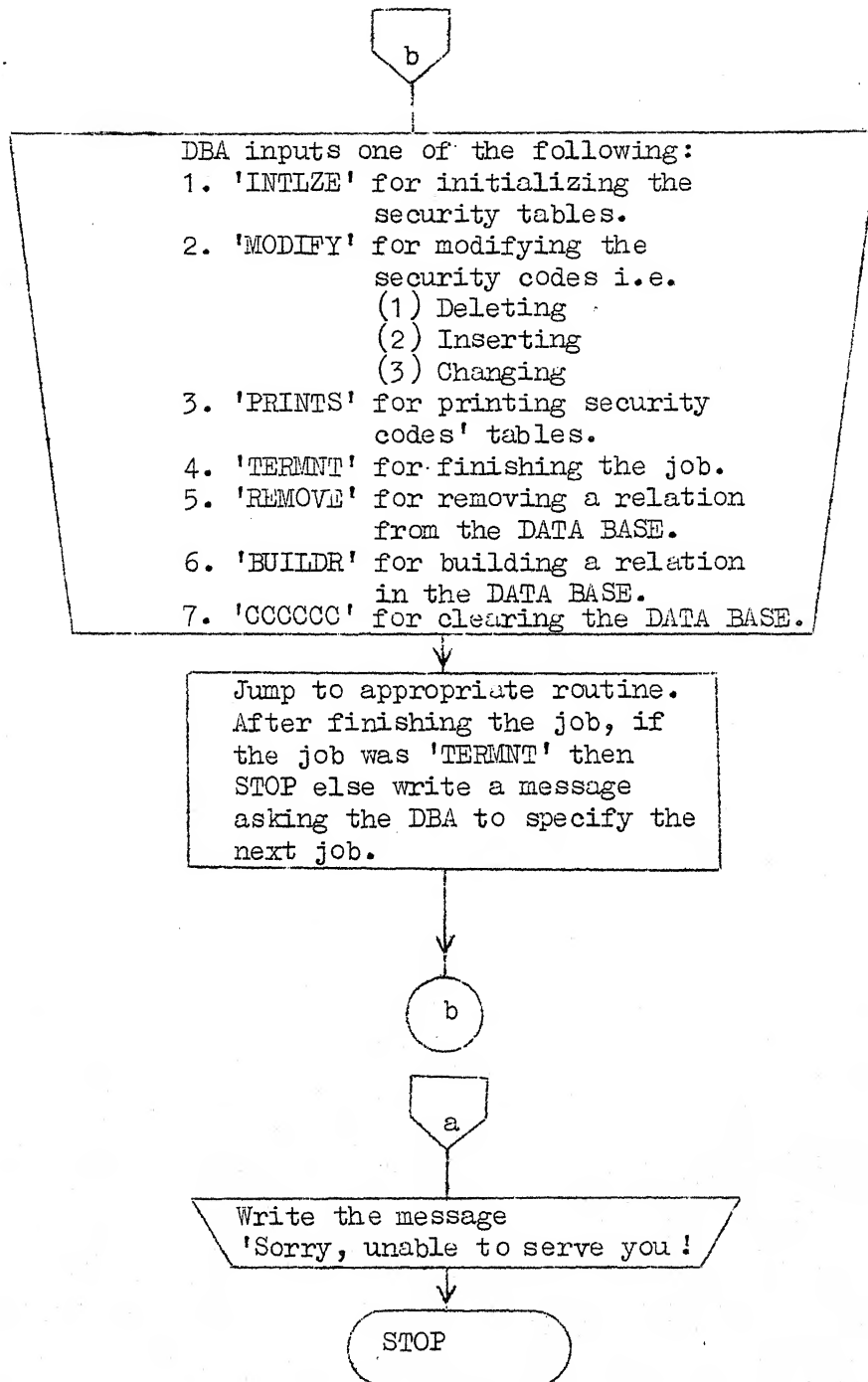


Figure 2-11: Flow chart for processing DBA's request.

shows the path that the DBA has to go through to carry out his task. The flowcharts for these tasks are detailed in later chapters in the following sequence.

Chapter 3 - Initialising the Security Tables

Chapter 4 - Modifying the Security Tables.

Chapter 5 - Printing the Security Tables.

Chapter 6 - Removing a Relation.

-

3. INITIALISING THE SECURITY TABLES

As described in the previous chapter, there are two types of security codes -

1. First Security Codes.
2. Authority Augmenting Security Codes.

Both these security codes involve two types of tables for internal storage as seen in last chapter which are -

1. Codes' Table Directory
2. Codes Table .

Codes' Table Directory gives pointer to the sector of the Codes Table in appropriate sequence depending upon the group of codes that the sector contains. Initialisation routine clears all the authority augmenting security codes from the tables and also clears all the first security codes except the first security code of the DBA which is retained to allow him continued access, for adding new security codes to initialized tables.

Initialization is essential when the DBA wants to change any of the routines WORBLE, VORBLE and DVORBLE etc., because now the internally stored security codes and surface and sector number in ASECDC will be computed using new routines and therefore the tables must be initialized and filled all over again. Also in this case, when the routines are changed, internally stored first security code of the DBA has also to be changed to the value produced by these new routine while acting on DBA's first security code and this is done at the same time when routines are changed.

Since all the security codes and authority augmenting vectors are stored on the disk, some space needs to be assigned on the disk for the purpose of storing these. Because maximum of $1+18+1+18+450 = 488$ sectors of security space may be needed, we reserve 5 cylinders (500 sectors) on the disk and call this reserved space as "Disk Security Space". To manage the disk security space, we store the status of each sector (free or full) in a bit and a 496 bit vector (an array of 31 words starting from the address BITMP) reflects the status of disk security space at any time. The routine ALOCTS allocates a sector of the disk security space, makes its status bit 1 and return the address of this sector to the calling program. The routine RELSEC releases a sector of disk security space with sector address given by the calling program by clearing its status bit. The program for these two routines appear in the Program Listings.

INITIALIZATION ALGORITHM

This algorithm in the form of a flowchart is shown in Figure 3-1. The equivalent program appears in the Program Listings. It deletes all security codes except DBA's first security code from the security tables. It also changes the disk storage addresses of FSECDC and ASECDC by allocating them new sectors and releasing the sectors previously held by them, thus making the positions of these tables changable. After initialisation, new security codes can be added using the MODIFY routine in the manner explained in the next chapter.

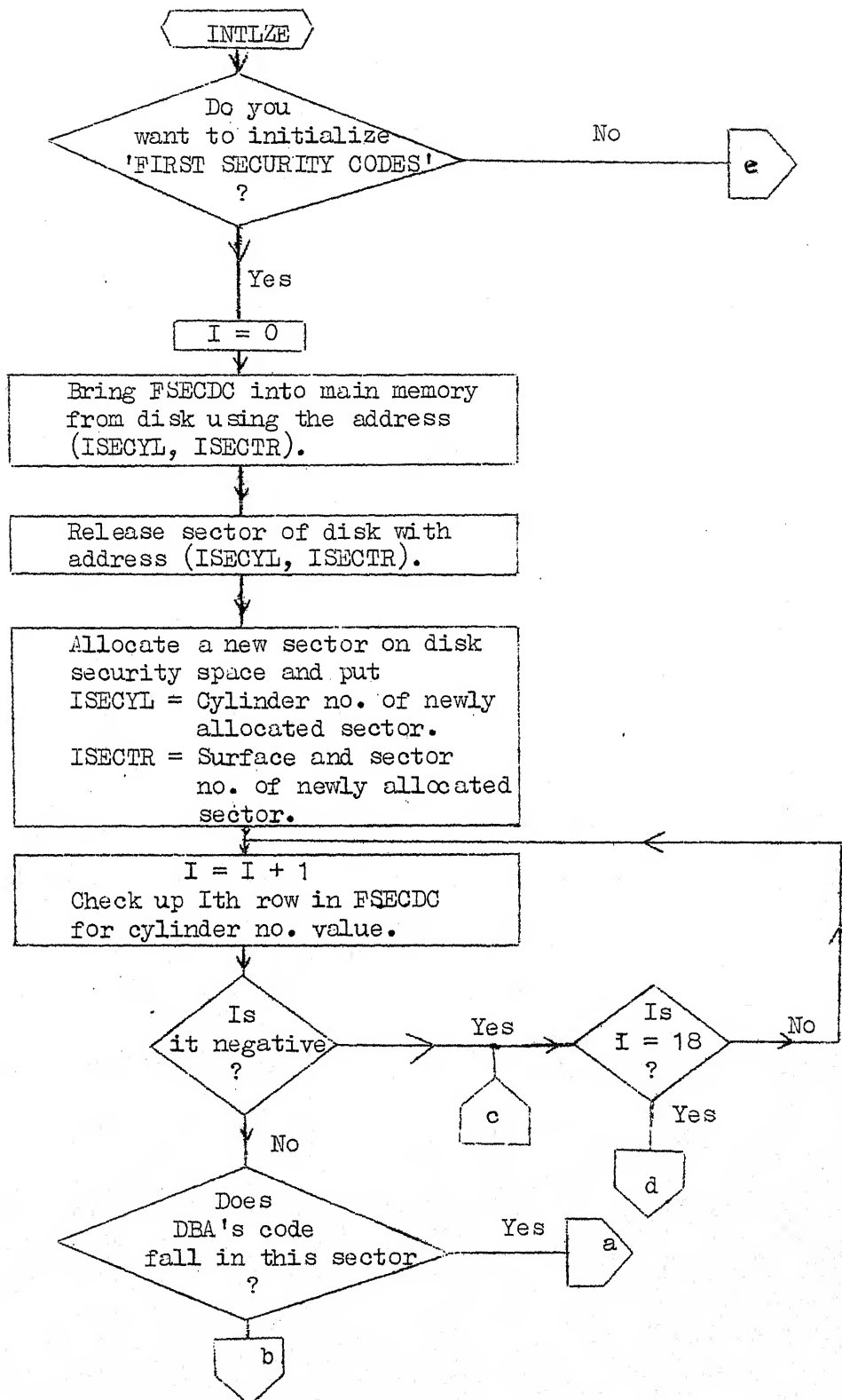


Figure 3-1: Continued on next page.

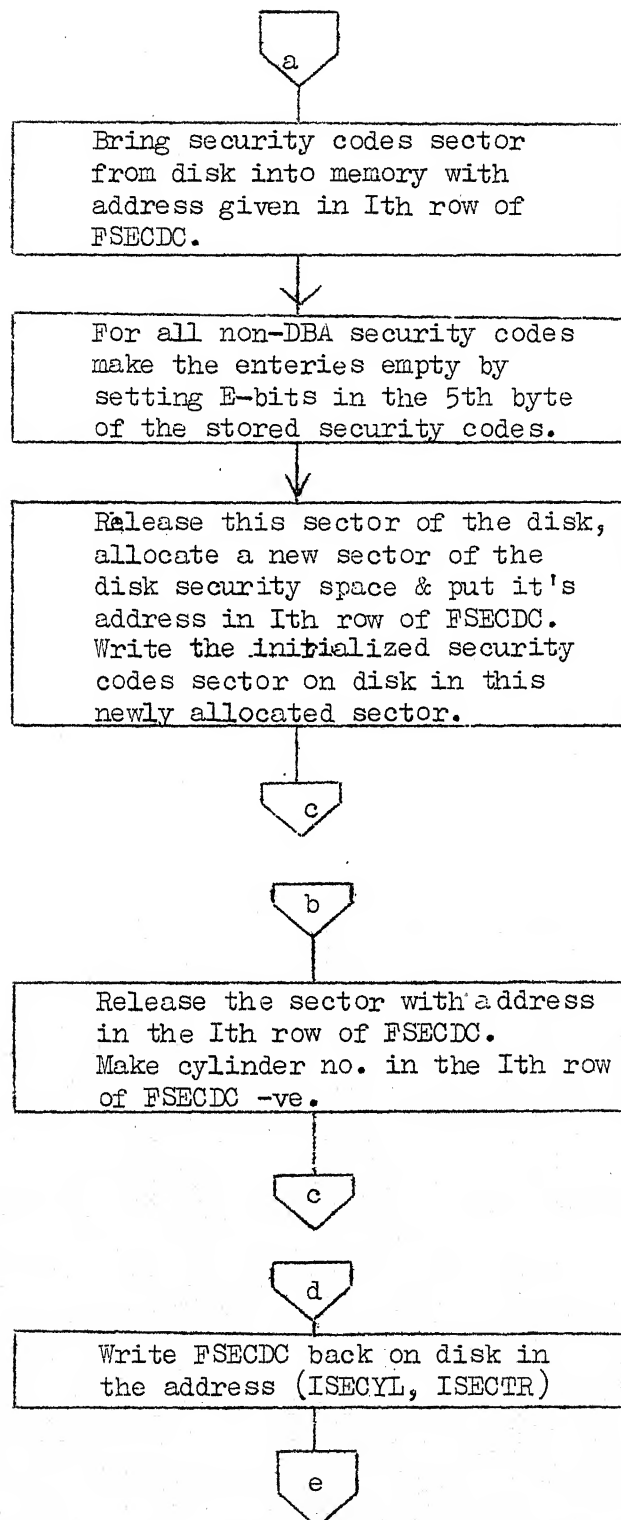


Figure 3-1: Continued on next page.

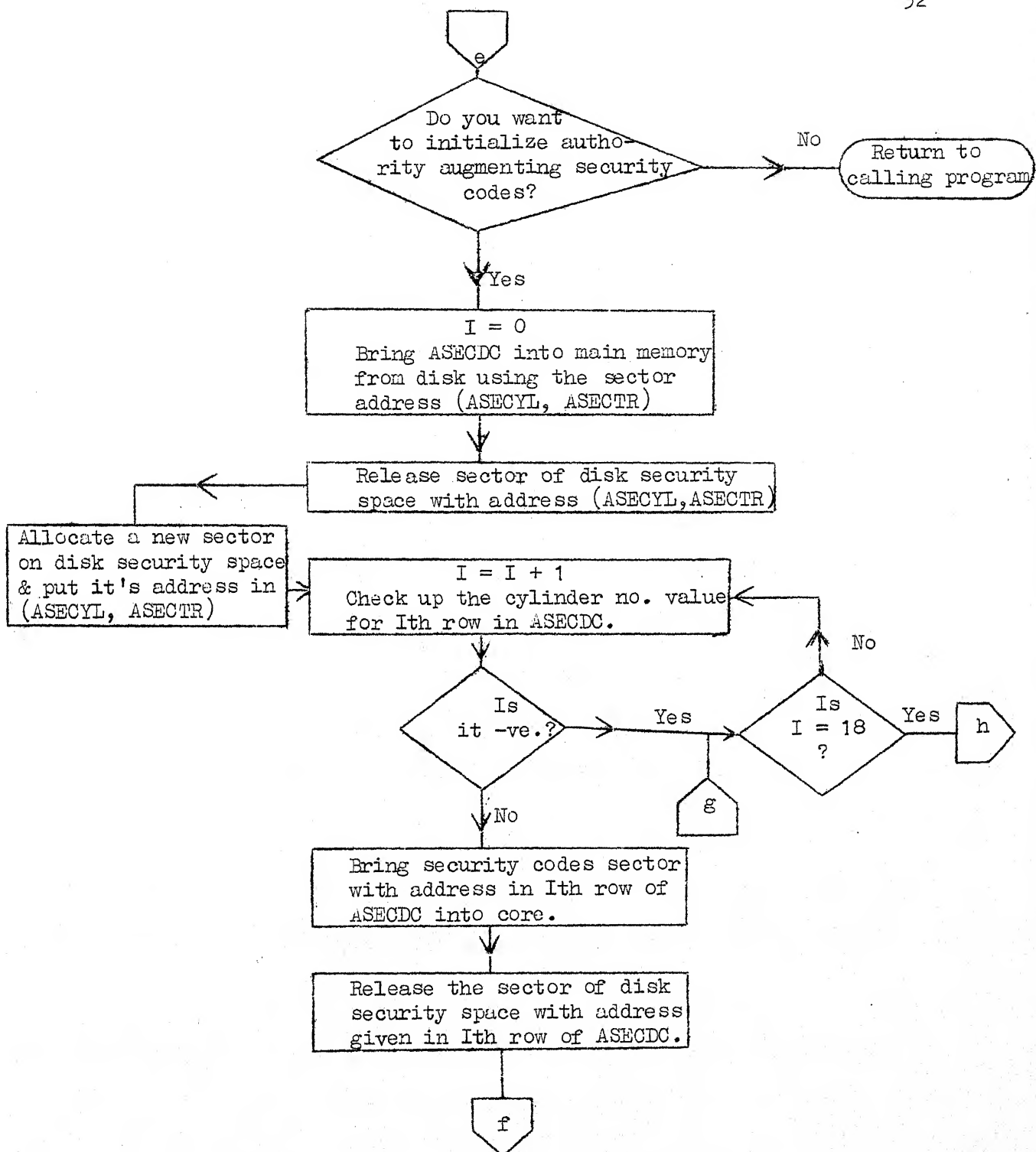


Figure 3-1: Continued on next page.

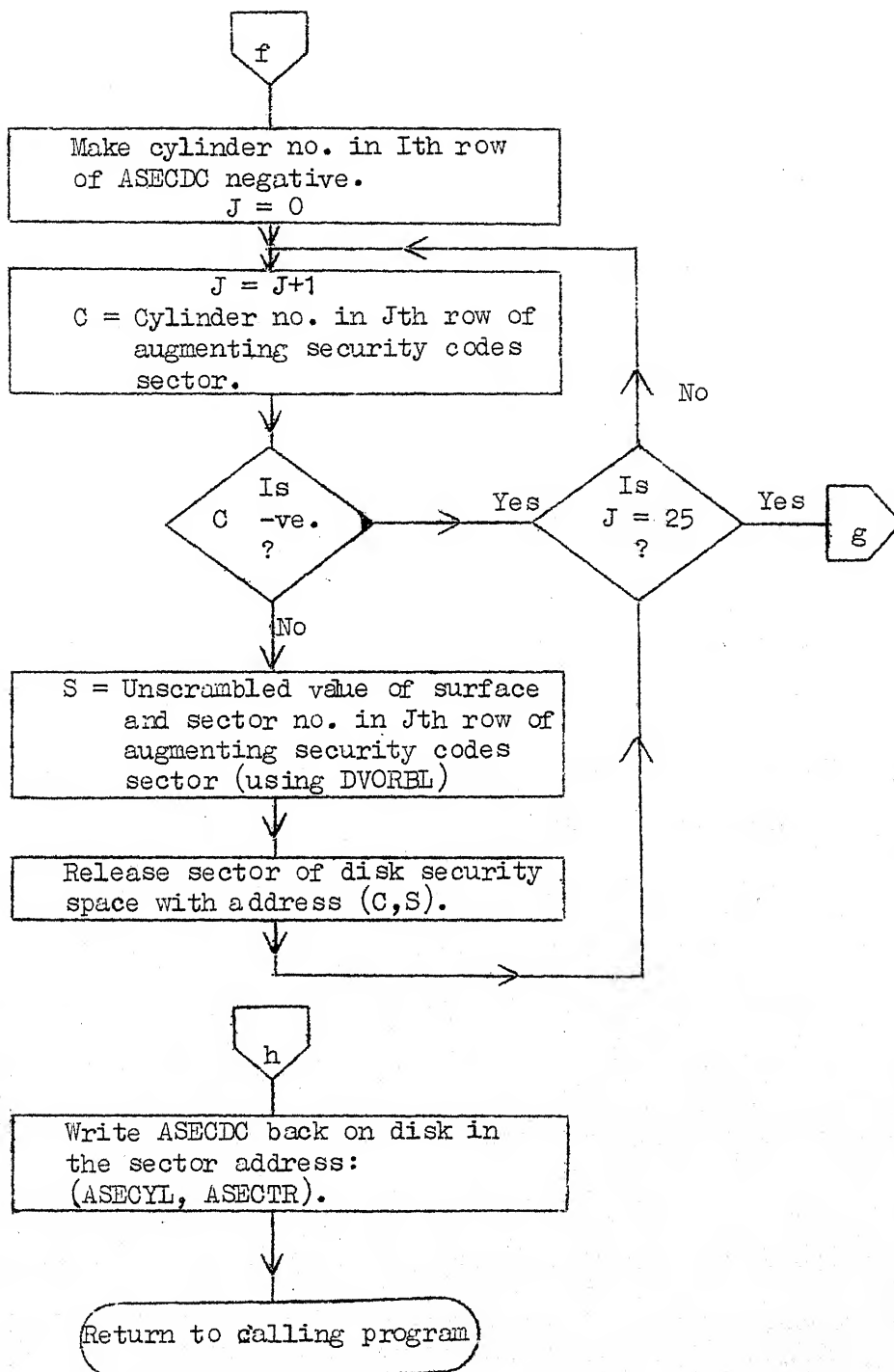


Figure 3-1: Flowchart for initializing the security codes.

4. MODIFYING THE SECURITY TABLES

Modification of the First Security ^{Code's} Tables can be done in any of the following ways:

1. Deleting a first security code
2. Inserting a new first security code or replacing an existing one.

Modification of the authority augmenting security code can be done in any of the following ways:

1. Deleting the security code (and hence also its associated authority augmenting vector).
2. Inserting a security code and its associated authority augmenting vector.
3. Changing an already existing authority augmenting security code while leaving its associated augmenting vector intact.
4. Changing the authority augmenting vector while leaving its associated security code (already existing) intact.

Deletion of any security code (of either type) can be done by specifying only its first three digits i.e., its position. The syntax of the language statements for deleting a security code is described below in terms of its fields:

First field (one ch)	= 'D' indicating deletion of security code.
2nd field (one ch)	= '1' if first security code is to be deleted = '2' if authority augmenting security code is to be deleted
3rd field (3 chars)	= '3 digit No.' indicating the position of the security code.
4th field (one char.)	= "." It is statement terminator.

For inserting a new security code or changing the value of an existing security code or authority vector, the syntax of the language statements is described below in terms of its fields -

First field (one character) = 'E' indicating insertion of new security code or authority augmenting vector or both.

Second field (one character) =

- '1' if first security code is to be inserted or changed.
- '2' if value of an existing authority augmenting security code is to be change leaving its associated authority augmenting vector intact.
- '3' if a new authority augmenting security code with its associated vector is to be inserted.
- '4' if the vector associated with an existing authority augmenting security code is to be charged while leaving the security code intact.

Third field =

- '13 digit security code' (for second field = 1,2 or 3)
- '3 digit No.' indicating position of security code (for second field = 4)

Fourth field =

- Authority vector with syntax explained below (for second field = 3 or 4)
- Null (for second field = 1 or 2)

Fifth field (one character) = '.' It is statement terminator.

The syntax of the authority vector (fourth field above) is expressed by the regular expression (~~N~~, (A,)*A)* where -

N represents byte No. of the vector from where to start in decimal.

A represents contents of corresponding bytes of authority vector in octal.

e.g., "#10,011,023,054 #55, 111, 321, 12" corresponds to the following contents of authority vector bytes.

<u>Byte No.</u>	<u>Contents in Octal</u>
1 to 9	000
10	011
11	023
12	054
13 to 54	000
55	111
56	321
57	012
58 to 256	000

Now we shall see examples for all types of statements alongwith their explanations:

Example 1: "D1 123"

This statement shall delete first security code whose first three digits are 123.

Example 2: "D2 421." : This statement shall delete authority augmenting security code whose first three digits are 421.

Example 3: "E1 159111111111." : This statement inserts the first security code "159111111111" in the set of valid first security codes. If a first security code starting with "159" digits already exists, the new security code takes its place and previous code is removed.

Example 4: "E2 126111111111." : This statement changes an already existing authority augmenting security code while leaving the associated vector intact. If no authority augmenting code with positional digits "126" exists, this statement is ignored and an error message is printed.

Example 5: "E3 021111111111 / 10,011,230 / 20, 11." : This statement introduces an authority augmenting security code "021111111111" in the list of valid codes with the associated authority augmenting vector as below:

<u>Byte No.</u>	<u>Contents in Octal</u>
1 to 9	000
10	011
11	230
12 to 19	000
20	011
21 to 256	000

If an augmenting security code with positional digits "021" already exists, then the value of this security code and its associated vector are replaced by new values of both.

Example 6: "E4 111 ~~/~~0,011,230~~/~~20,11." : This statement changes the authority vector associated with the authority augmenting security code whose positional digits are "111" from present value to that shown in Example 5 above, while leaving the security code value intact. If no security code with positional digits "111" exists already, this statement is ignored and an error message is printed.

Any number of these modification statements can be given at a time through the input device which can be a card reader ("CA"), high speed paper tape reader ("PA") or keyboard ("KY"). More than one statement can be put on one card and a single statement can be continued on many cards. All columns (1-80) are usable. There can be arbitrary number of blanks between various fields of a statement and between statements. To indicate that no more statements follow, we put the character "F" after period of the last statement.

MODIFICATION ALGORITHM

Assumptions made -

1. GET routine is available. This routine when called puts the next character of the input in the variable "CH ". When calling this routine for the first time in the program, we clear the variable FLAG of this routine to indicate fresh input. Also the variable INPUT is loaded with the device number of the input device before calling this routine.
2. GET.3(N) routine is available which reads the 3 characters (digits) of input and puts their binary value in N.
3. GET.10(N1) routine is available which reads ten characters from input and puts them in a buffer starting from location N1.
4. WORBLE (N1,N2) routine is available which generates a ten digit number from two 10 digit number inputs from buffers N1 and N2 and puts the generated result in buffer N1.
5. GIVSEC (N1,N2) routine is available which takes 10 byte (digit) input from the buffer N1 and puts its 4 bytes binary equivalent in buffer N2.
6. AUTHSC routine is available which reads the authority vector part of the input modification statement and builds its equivalent authority vector in a specified area of the core.

The program for all the six routines above appear in the Program Listings.

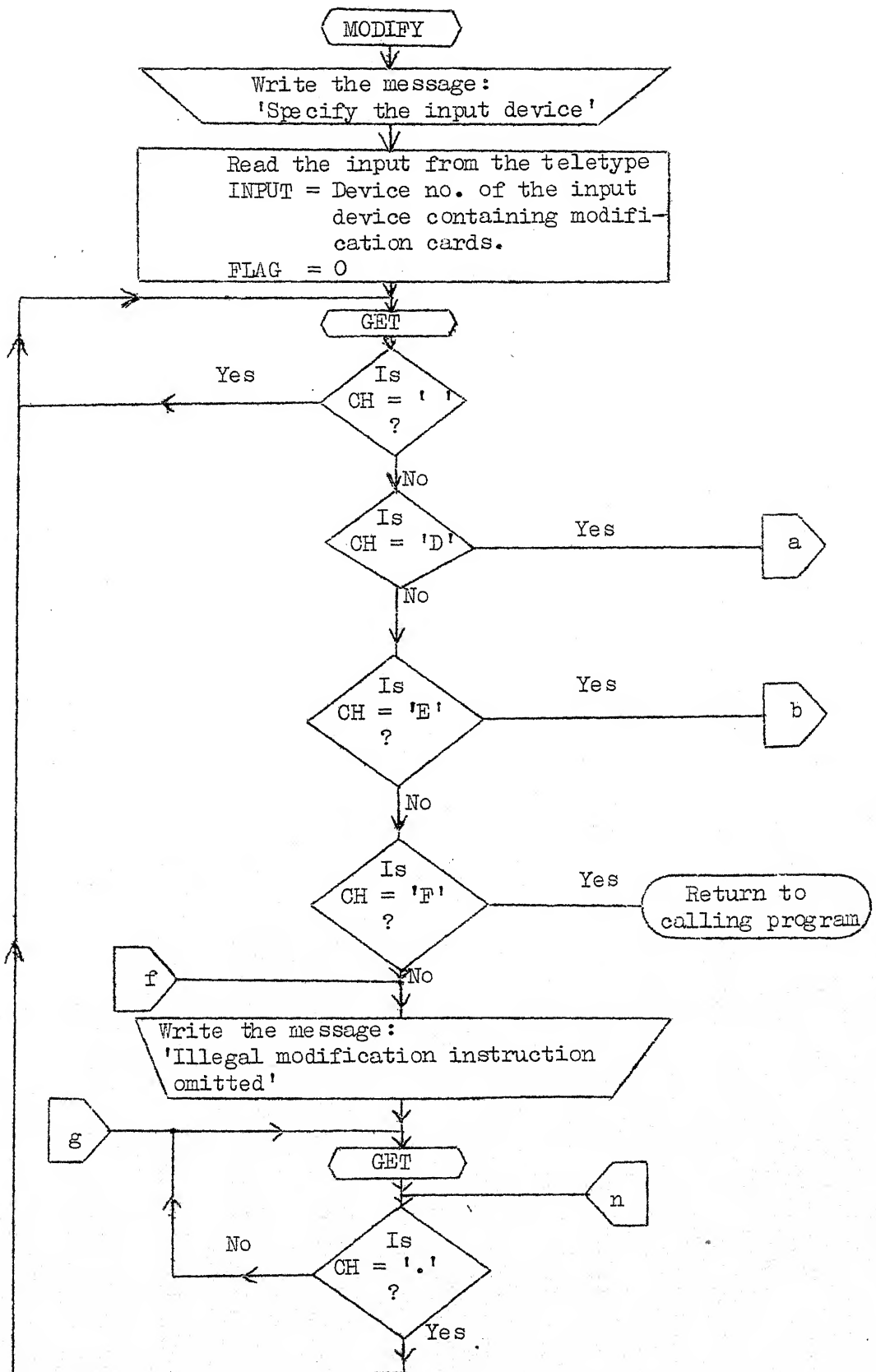


Figure 4-1: Continued on next page.

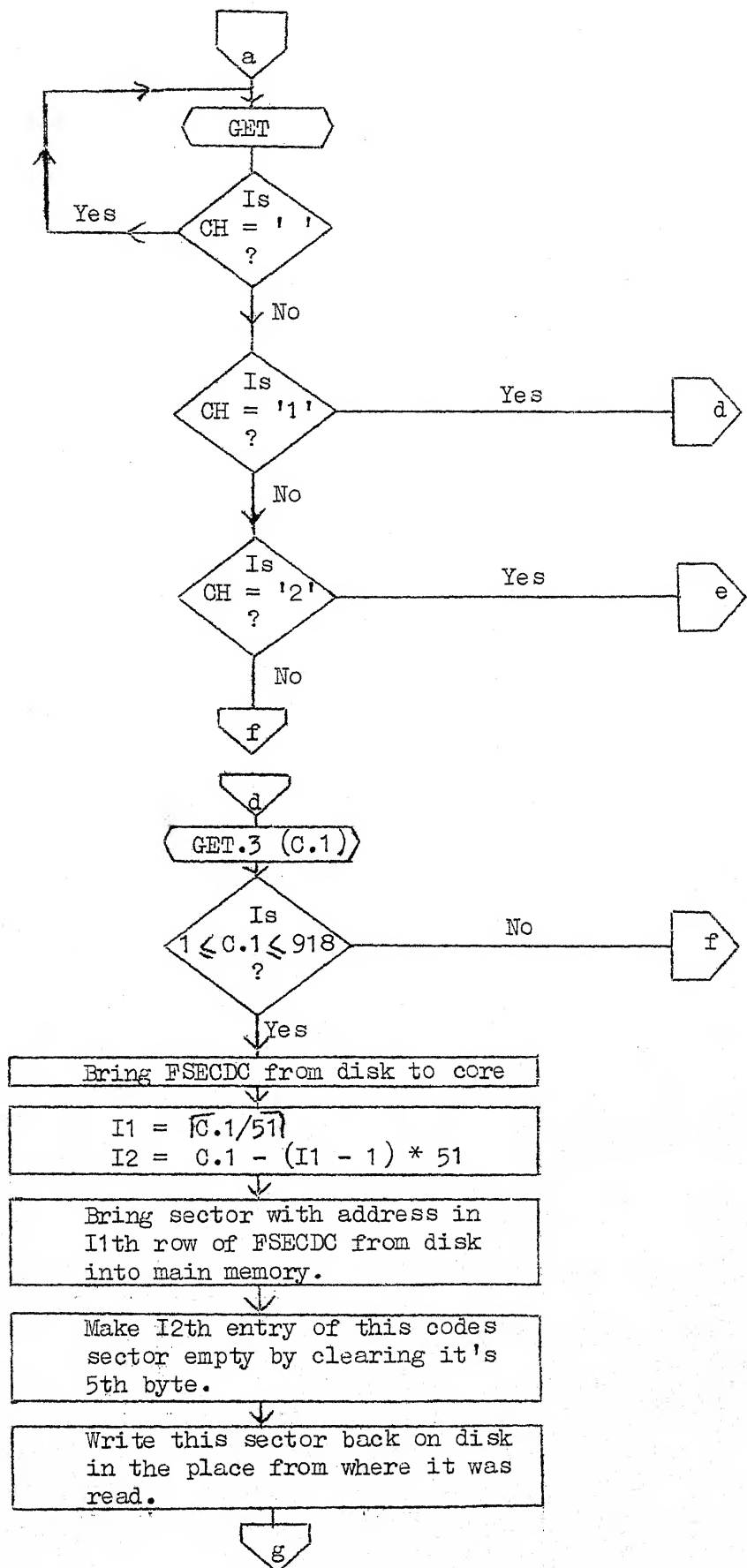


Figure 4+1: Continued on next page.

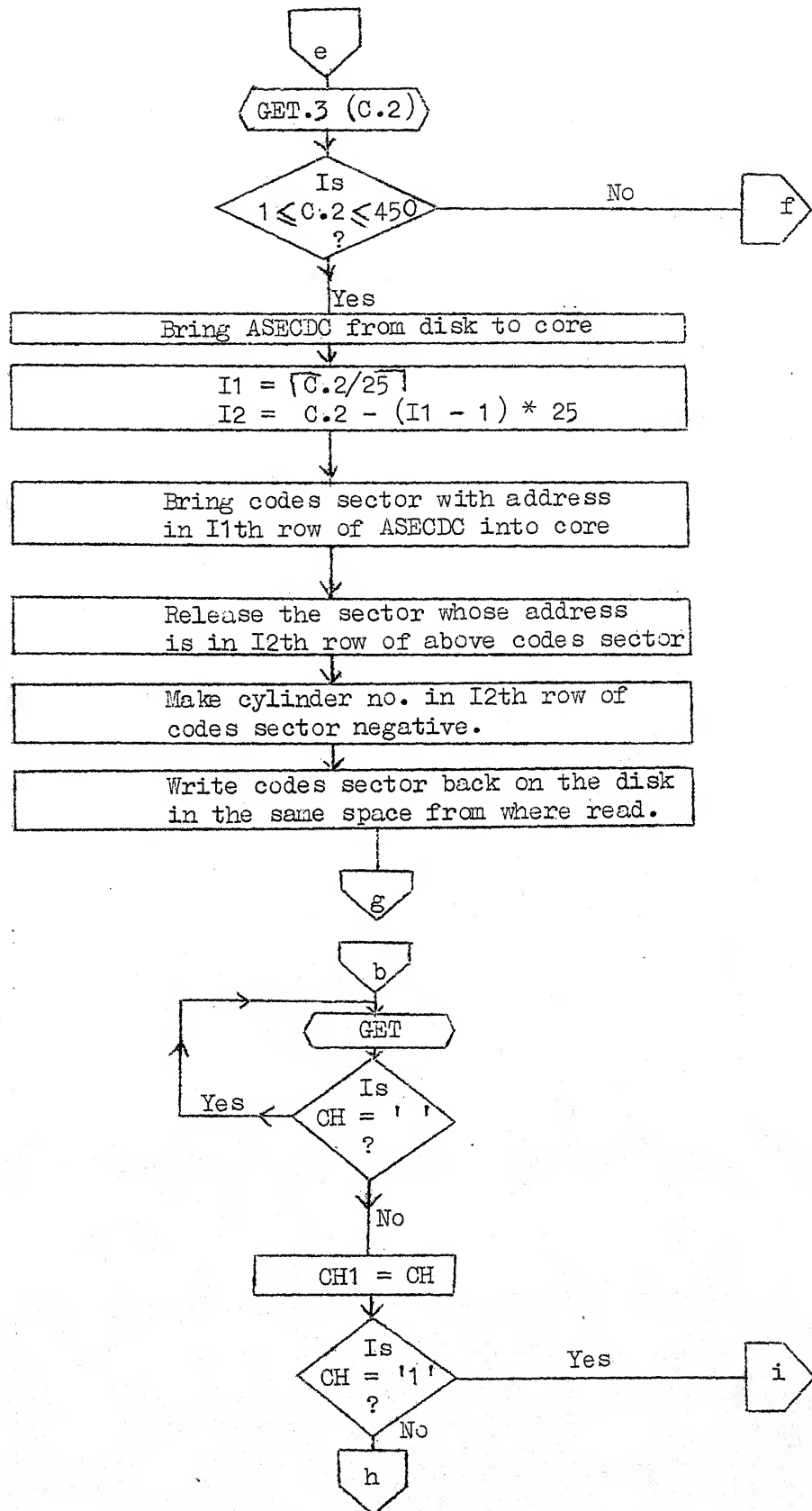


Figure 4-1: Continued on next page.

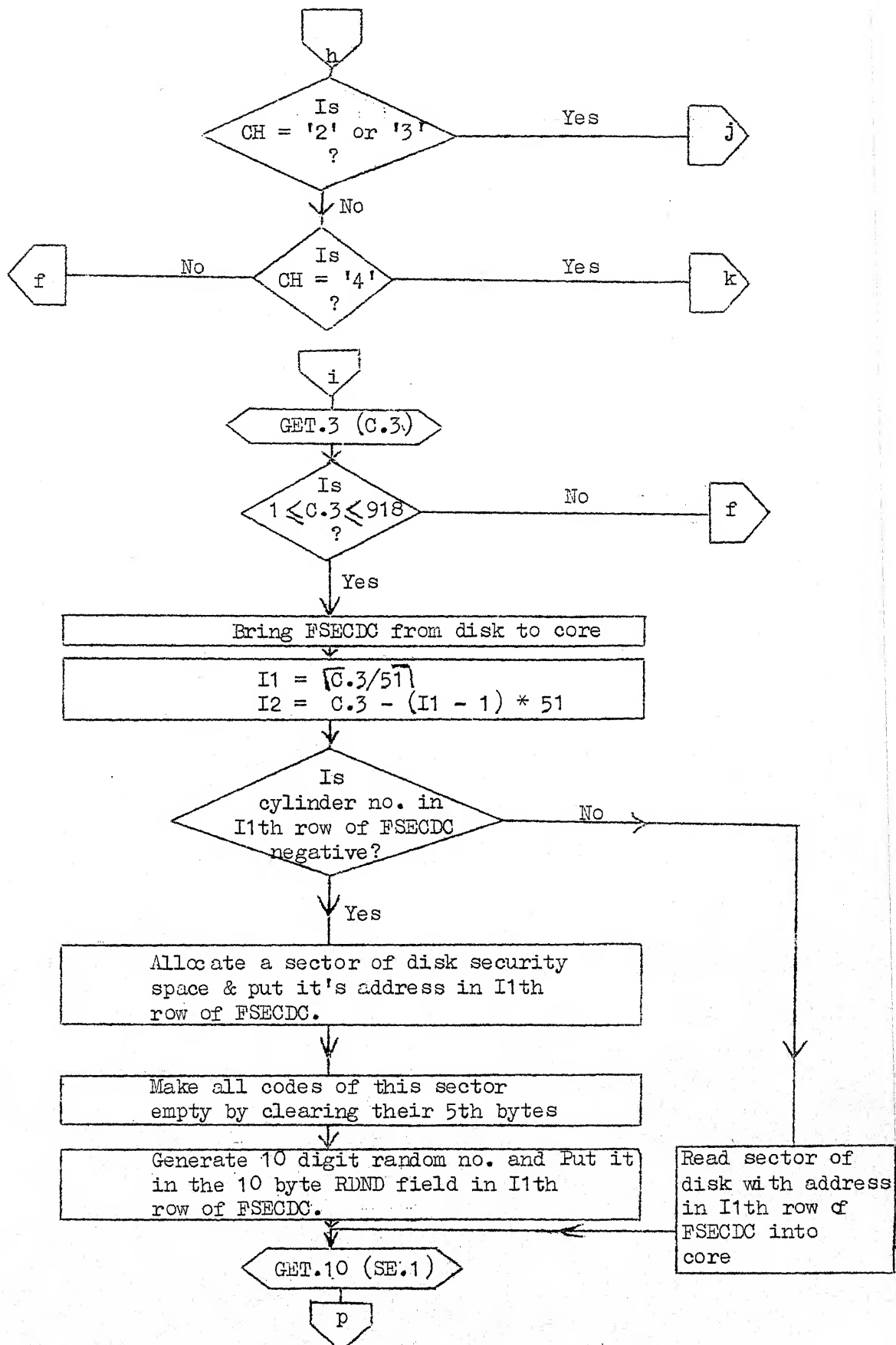


Figure 4-1: Continued on next page.

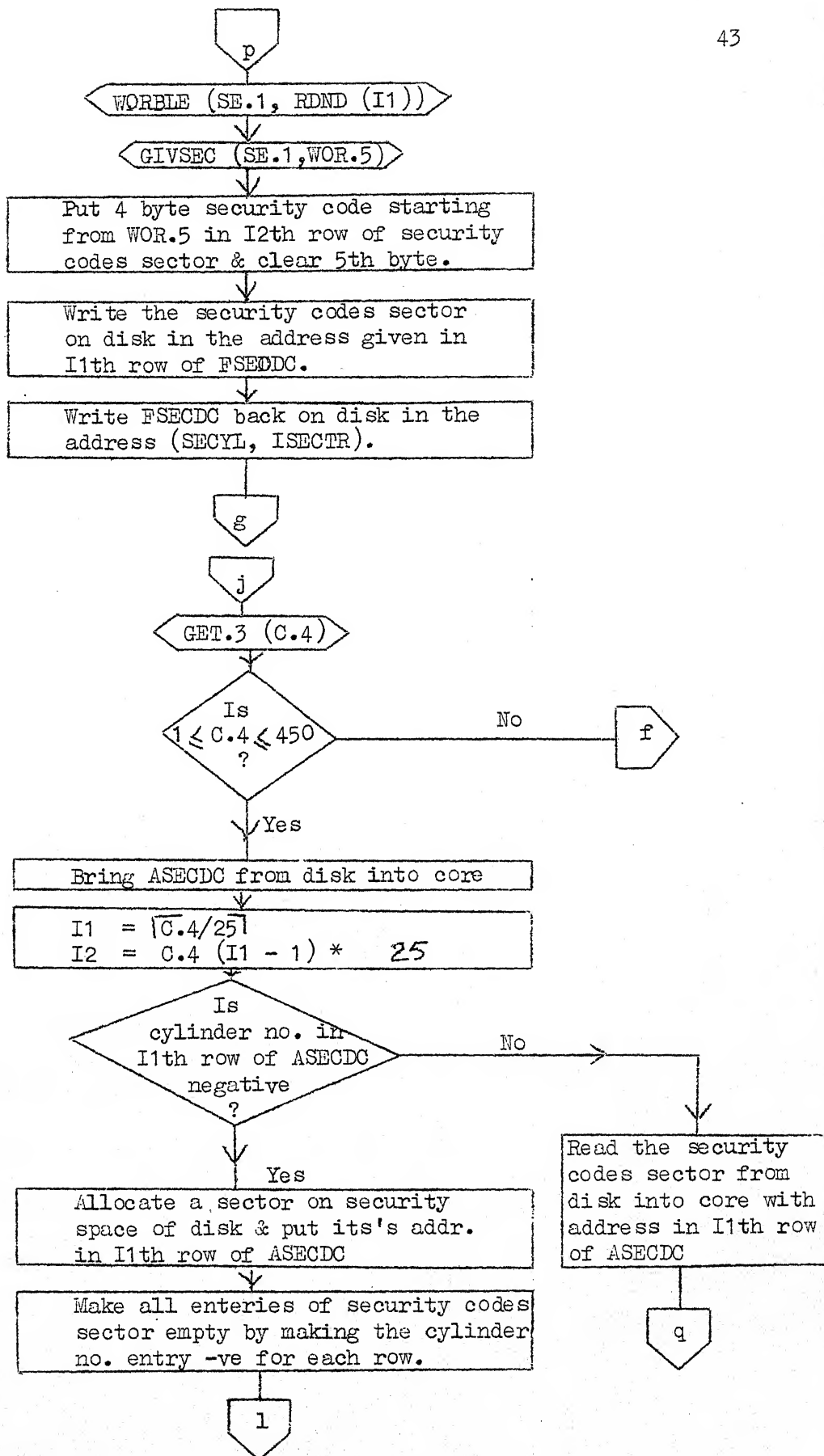


Figure 4-1: Continued on next page.

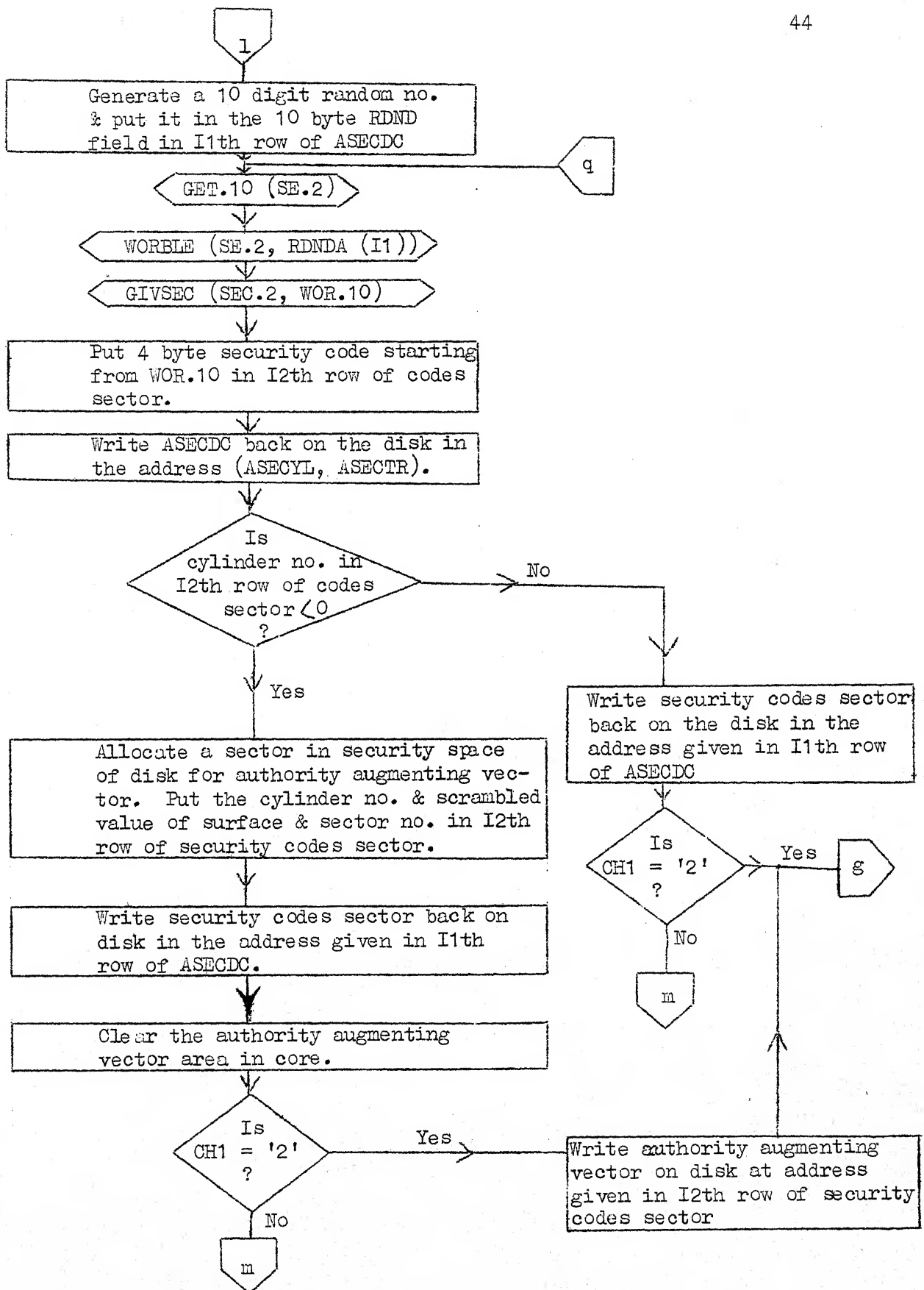


Figure 4-1: Continued on next page.

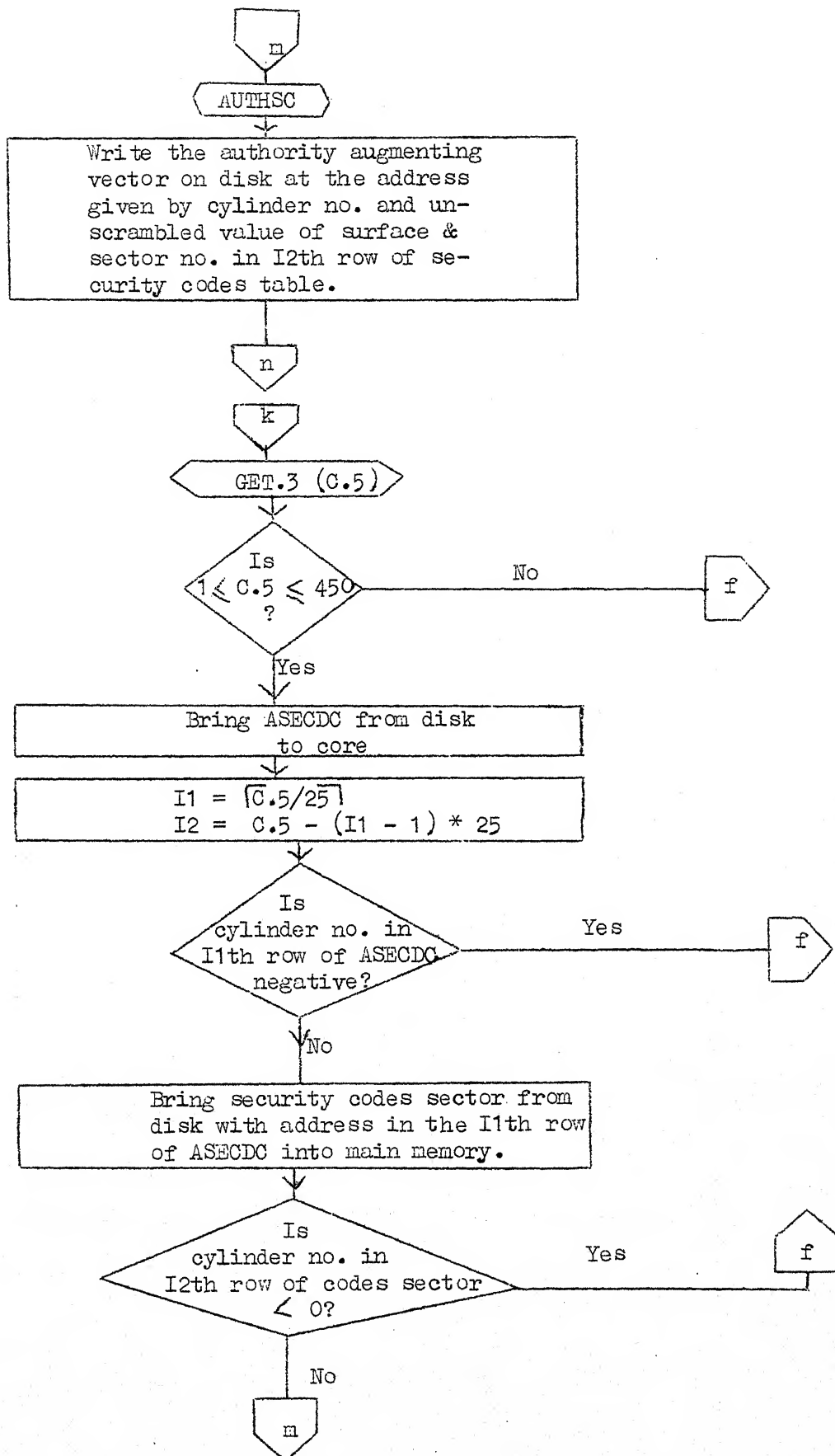


Figure 4-1: Flowchart for modifying the security codes' tables.

The modification algorithm in the form of a flowchart is shown in Figure 4-1. The program for this algorithm appears in the Program Listings appearing at the end of the thesis.

After modifying the security tables, the DBA can print the tables to satisfy himself that proper modification has been carried out. This he can do using the PRINTS routine which is given in the next chapter.

-

5. PRINTING THE SECURITY TABLES

ALGORITHM FOR PRINTING

The data base administrator may be interested in knowing the status of security tables at any time. The "PRINTS" routine gives the DBA the facility of printing FIRST Security codes' tables and Authority Augmenting Security Codes Tables. First the DBA is asked to specify the output device where the tables are to be printed. The DBA gives "PR" for line printer and "TT" for teletype. Thereafter, system asks the DBA if he wants to print "First Security Codes' Tables". DBA gives "Y" "If", if he wants to print the tables. Otherwise he gives "N" "If". Similarly the DBA is asked if he wants to print "Authority Augmenting Security Codes' Tables". Again he gives "Y" "If". for printing the tables and "N" "If." for not printing the same.

In case of First Security Codes, first the system prints "First Security CodesTable Directory", i.e., FSECDC in the format of Figure 2-4. Then the system prints each existing first security code along with its status. Status can be any of the following -

- (i) A blank status field indicates that the code has been used properly.
- (ii) "ILLEGLE" in status field indicates that the user tried to give wrong authority augmenting code to access the system illegally.
- (iii) "MORASK" in status field indicates that the user tried to ask the system for more than permitted to him, by giving such retrieval or update commands as were not permitted to him.

In case of Authority Augmenting Security Codes, the system first prints "Augmenting Security Codes Table Directory" i.e., ASECDC in the format of Figure 2-8. Thereafter, the system prints each existing authority augmenting code alongwith its associated authority augmenting vector. It also prints the sector address where this authority vector is stored on the disk. Since at any time, the data base shall generally contain much lesser number of fields then the maximum 256 permitted, the PRINTS routine asks the DBA to give a three digit input number (say N) specifying number of fields in the system. Then it prints only the first N fields of each authority vector. Moreover, if the DBA does not want to print any authority vectors but only the authority augmenting security codes, he may give $N=0$.

The flowchart of the algorithm for printing security codes appears in Figure 5-1. The equivalent program of this algorithm appears in the program listings.

TERMINATING THE JOB

Before terminating DBA's job, we must write all the variable data as outlined in Flowchart 5-2 back on the disk. So at the end of any operation that the DBA may perform on the data base, he must perform the job "TERMNT". With this, all useful information in core gets stored back on the disk and may be fetched for later usage.

Having described all the programs that go into building the security system for the data base, the only function left is removing a relation from the data base which is the subject matter for the next chapter.

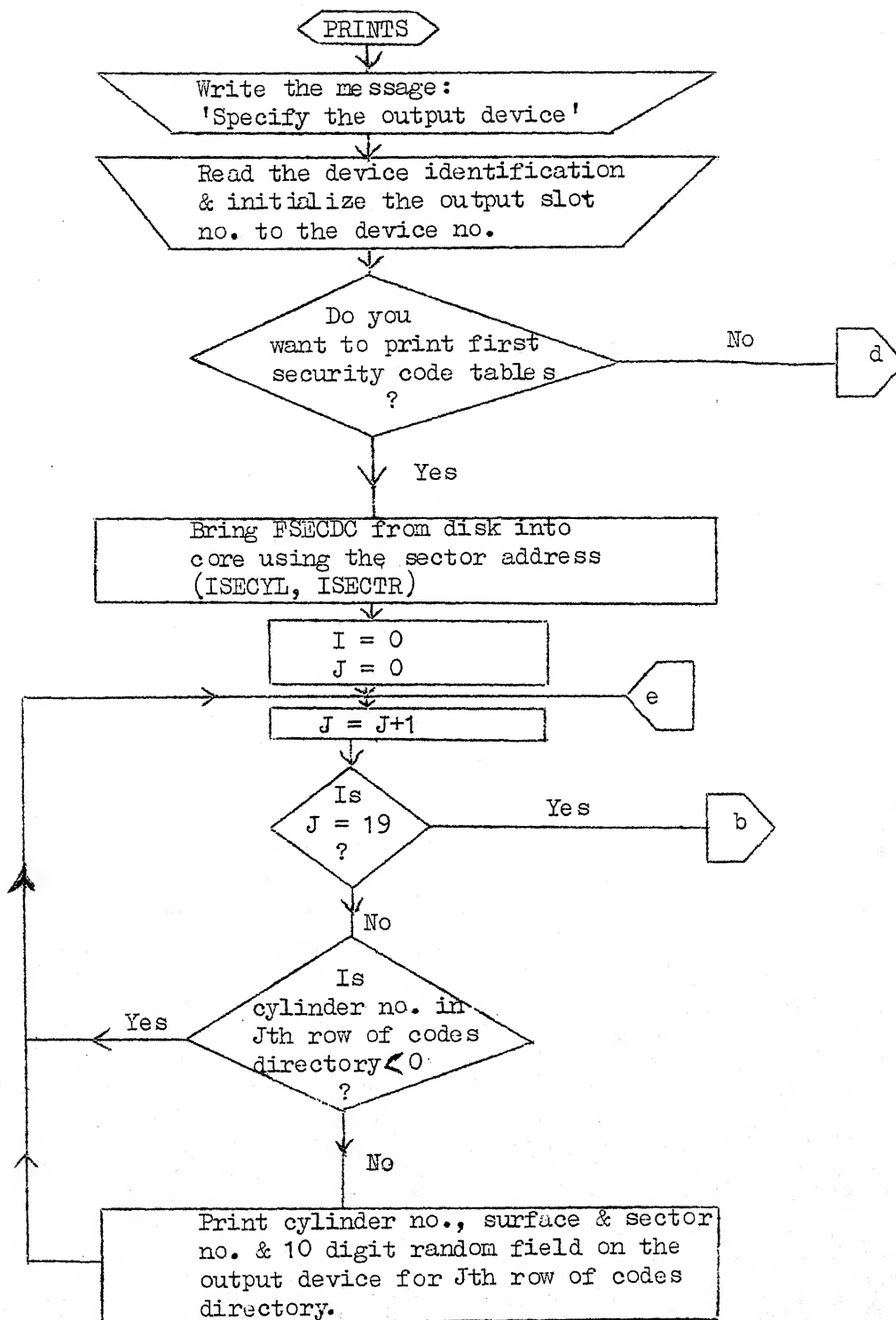


Figure 5-1: Continued on next page.

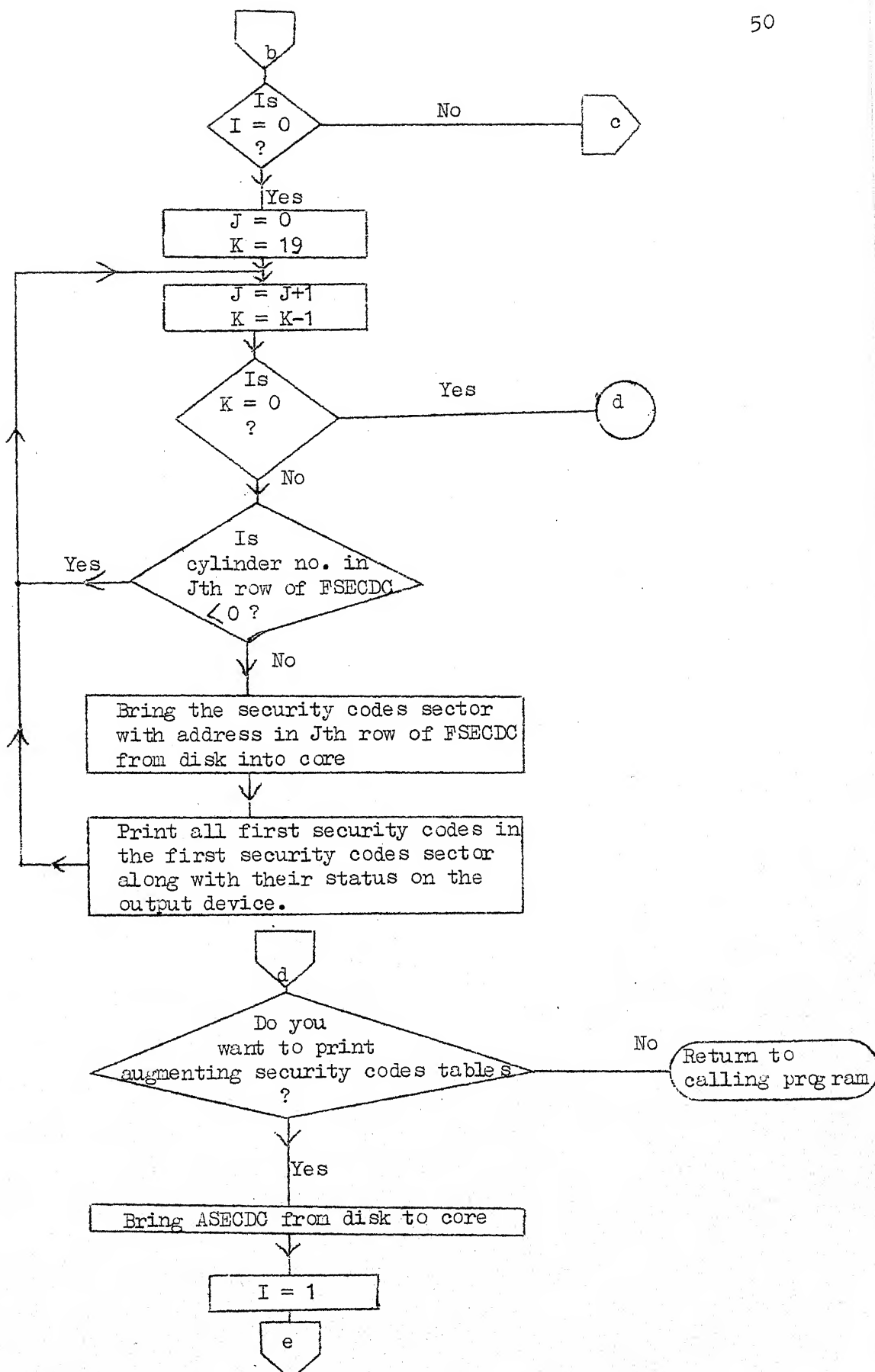


Figure 5-1: Continued on next page.

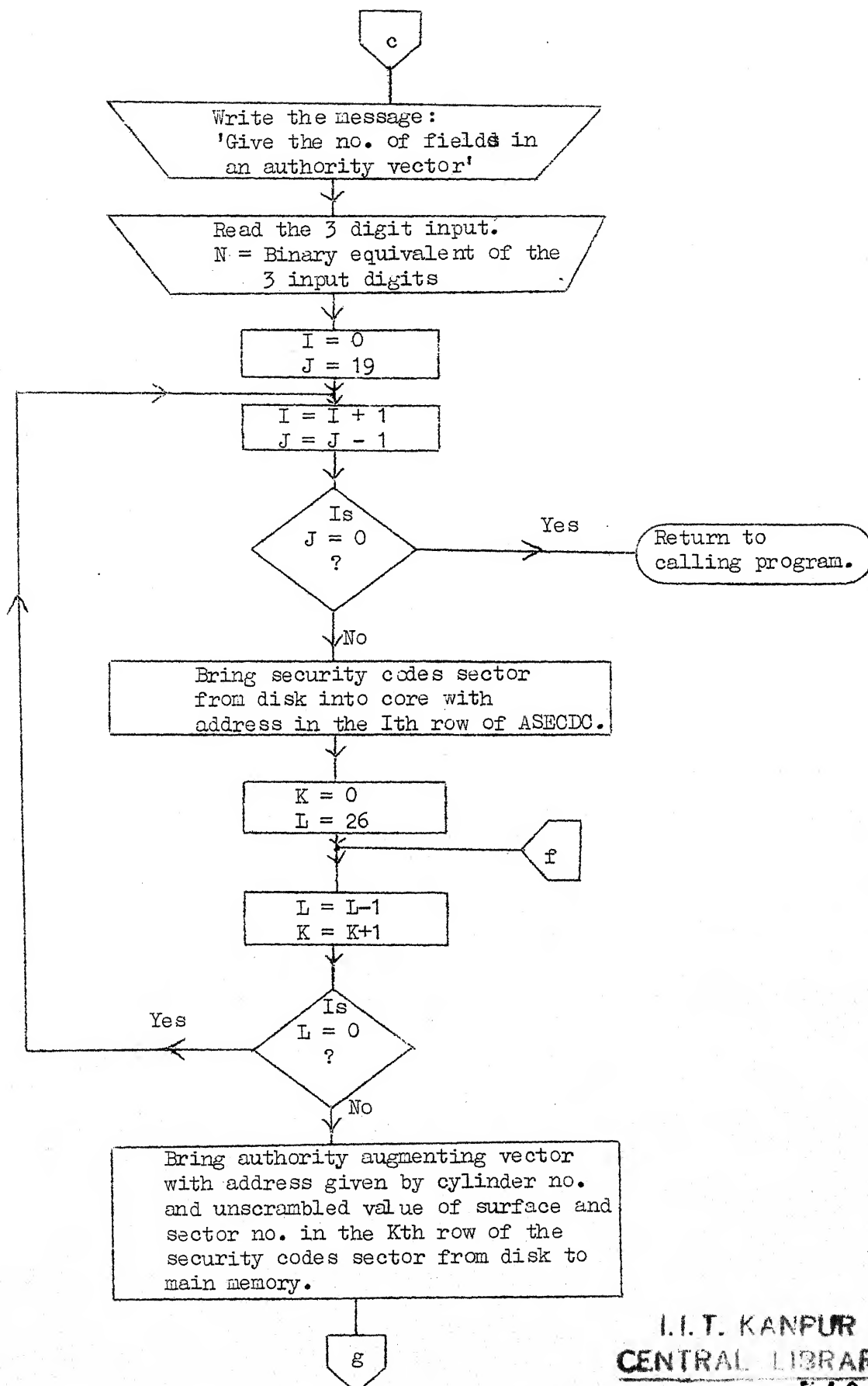


Figure 5-1: Continued on next page.

I.I.T. KANPUR
CENTRAL LIBRARY

Acc. No.

54948

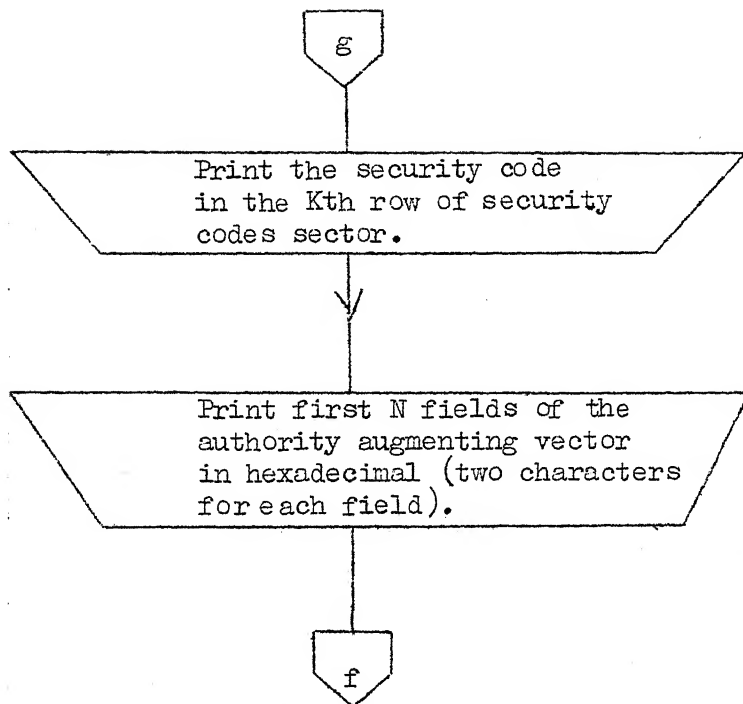


Figure 5-1: Flowchart for printing the security codes tables.

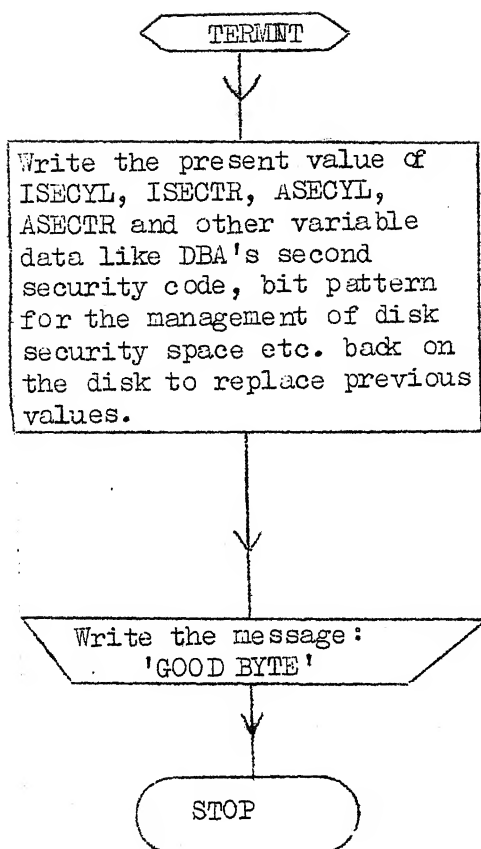


Figure 5-2: Flowchart for terminating DBA's job.

6. REMOVING A RELATION

As a data base grows older with time, it may become necessary to add more relation to the data base which may pertain to some data which has currently acquired importance, or to remove some relations which pertain to the data which has lost its relevance, or to re-organise the data base by combining some relations into one or breaking one relation into many or any combination of these in order to improve the efficiency of the system in the light of the past queries.

To illustrate by example let us say that we have a data base in IIT Kanpur where we store one relation for each year of students. Now as a new batch enters, a new relation shall have to be added to the system. But the relation corresponding to the outgoing batch may be dumped on some tape file (for any possible emergency need) and then removed from the data base.

Adding a relation is done by BUILDR whose programs appear in a separate thesis (Ref. 5). The present chapter discusses removing a relation. Since all the data structures used here appear in the thesis describing BUILDR programs, it is essential to go through it before studying this algorithm.

REMOVAL ALGORITHM

Removing of a relation from the data base involves the following tasks -

1. Removing this relation from the Relations Directory RELDIR by clearing the RELID field for this relation, so that this relation identification number can be used to define new relation.

2. Removing the field records corresponding to this relation from the field list "FDLIST" and closing this gap by moving the field records lying below this gap upwards.
3. Setting the cylinders occupied by the relation on the disk free, by clearing their status bits in the bit vector (BITMAP).
4. Clearing the entries corresponding to this relation from the primary index table and moving the other entries up to close the gap.
5. For all existing relations whose primary index entries have been moved up, reflecting this movement in the relations directory RELDIR by subtracting the length of the gap (i.e., length of upward movement) from their PMINX field values.
6. Writing these new tables values of RELDIR, FDLIST, primary index etc. back on the disk.

All these steps are reflected in the flowchart of removal algorithm shown in Figure 6-1. The flowchart uses a routine RELCYL(N) which releases cylinder No. N of the disk and returns control ^{to} the calling program. Flowchart of this routine has been drawn in Figure 6-2. Programs corresponding to these two flowcharts appear in the Program Listings.

A FEW WORDS ON REORGANIZATION

Reorganization can be of two kinds:

1. Reorganizing the data base by merging two or more relation or breaking one relation into many or a combination of both to build new relations. This can be done as follows:

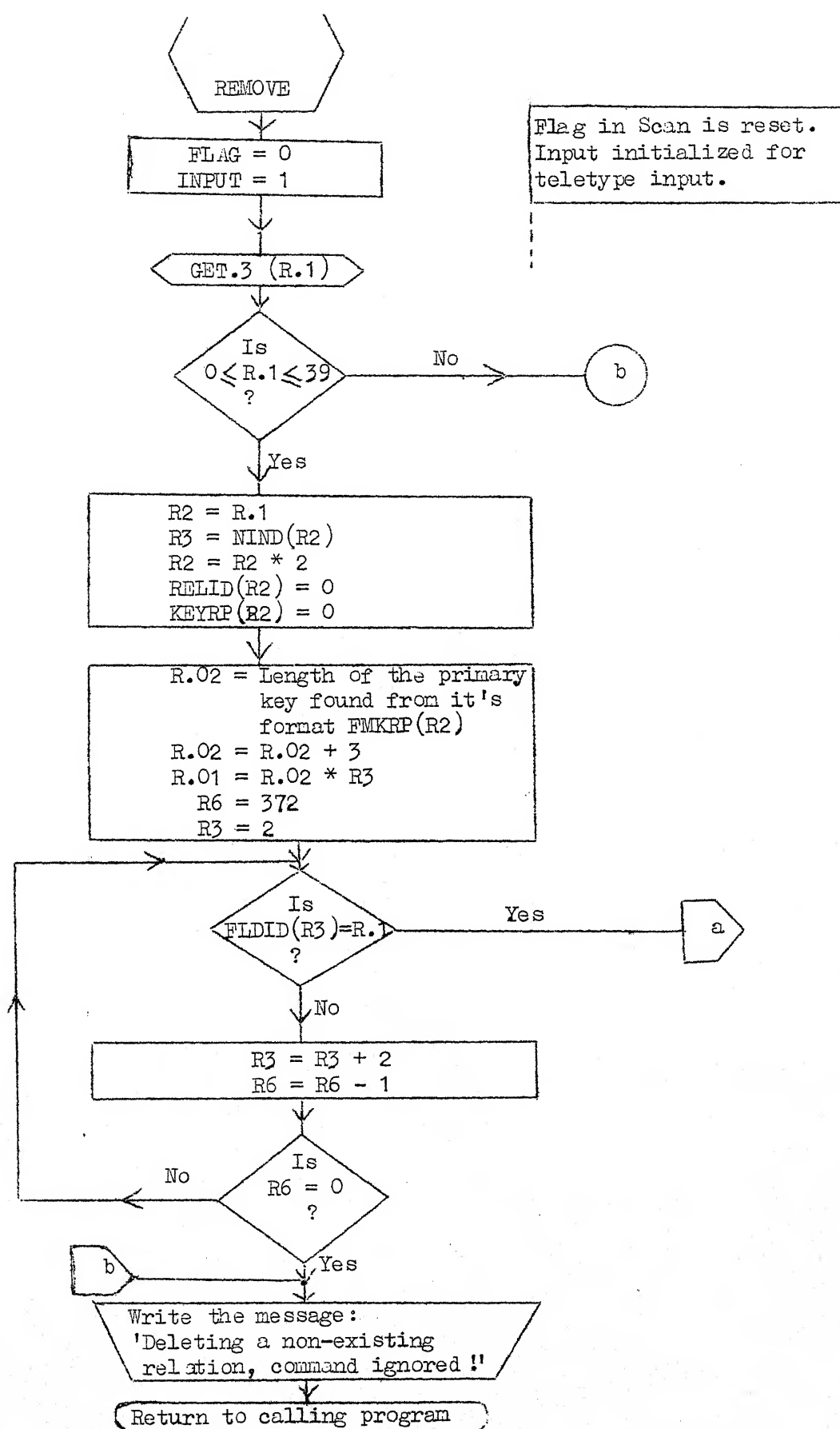


Figure 6-1: Continued on next page.

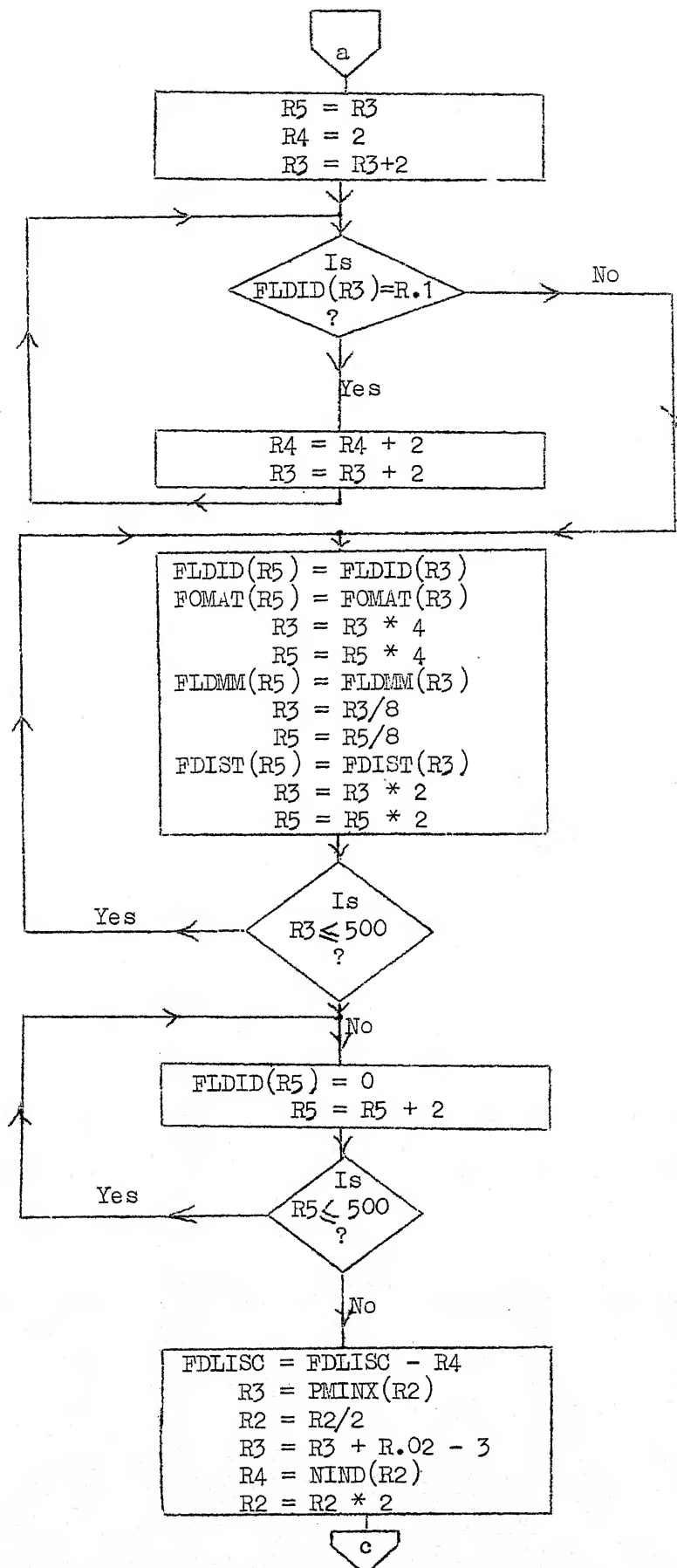


Figure 6-1: Continued on next page.

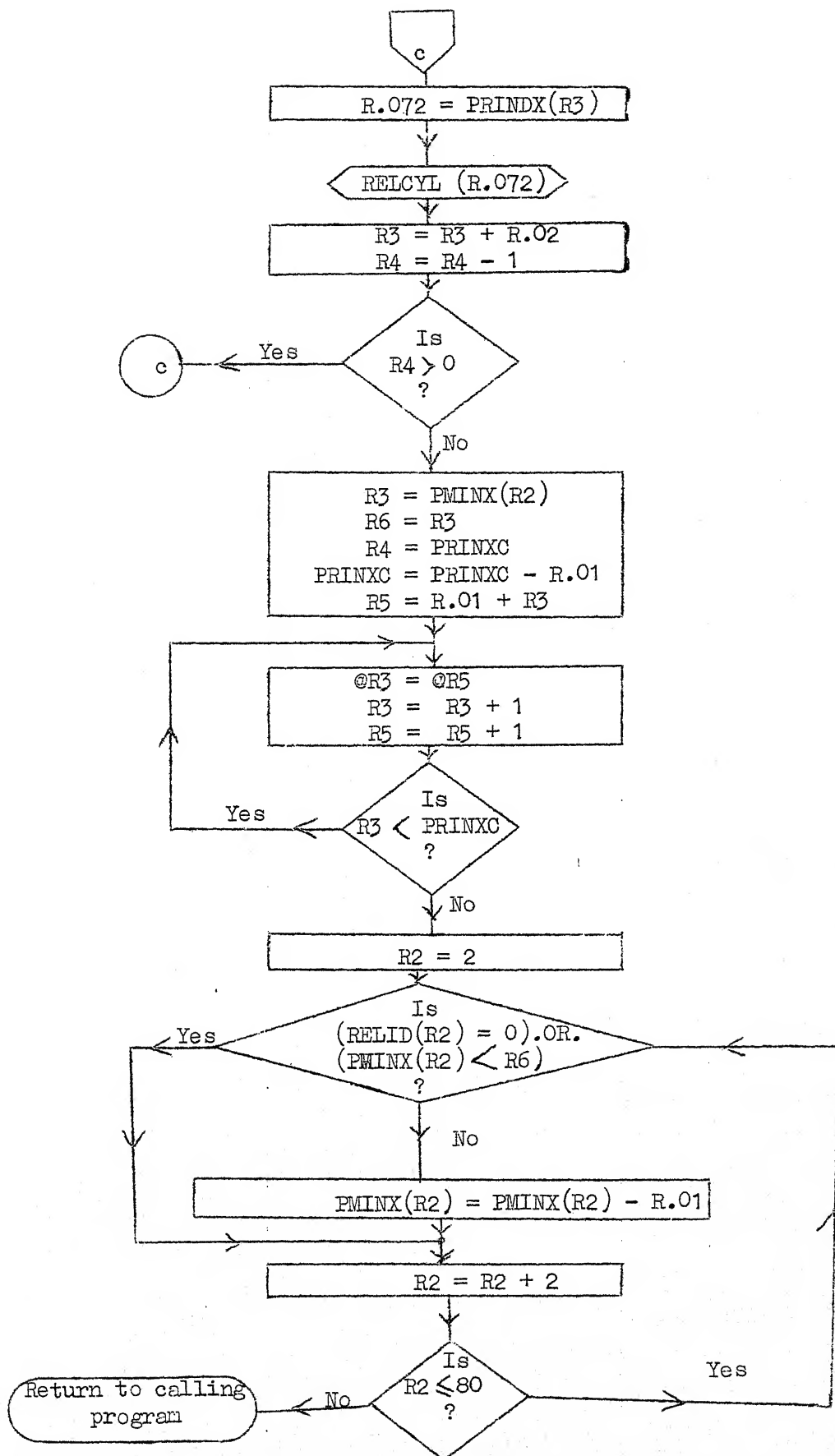


Figure 6-1: Flowchart for removing a relation from the data base.

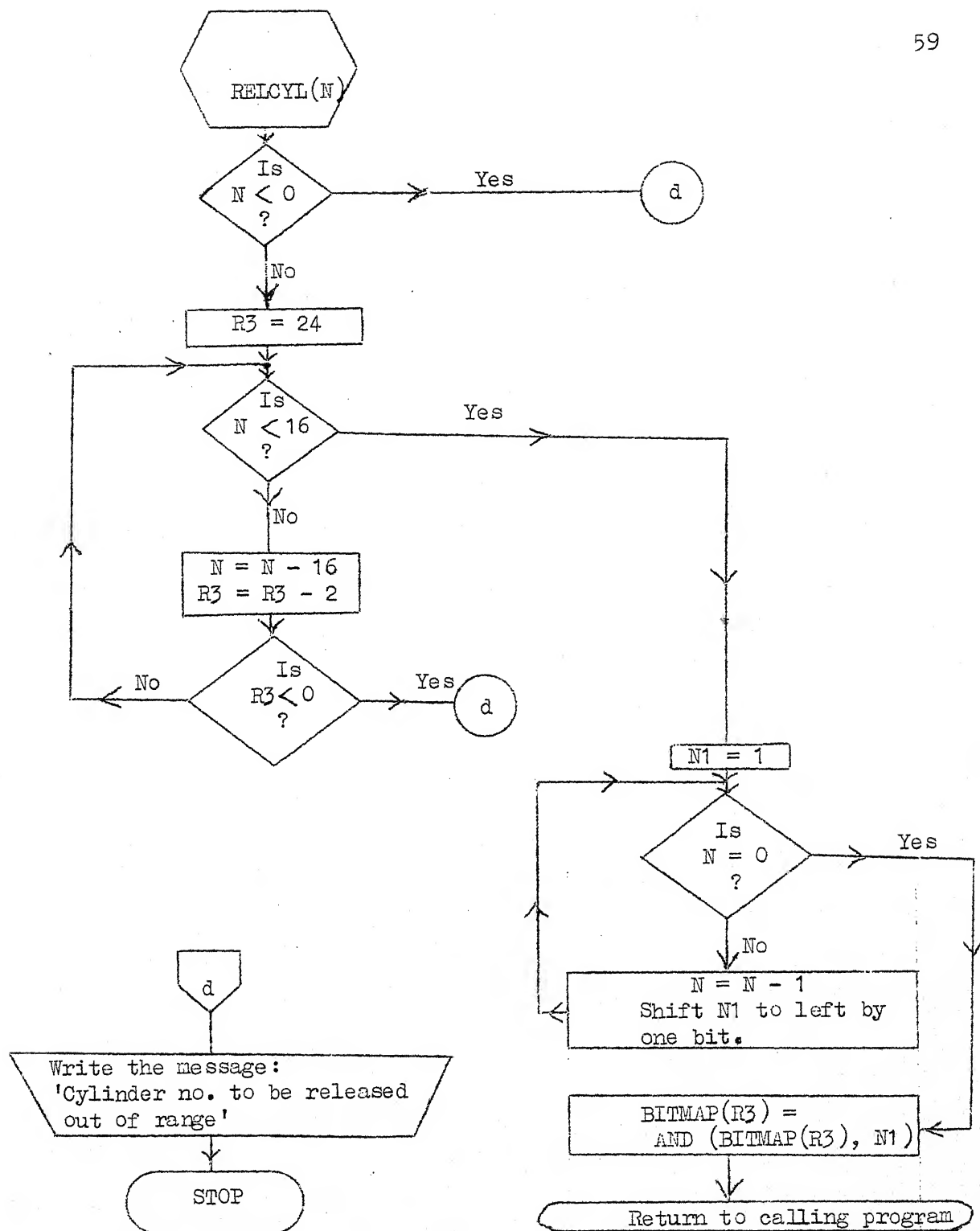


Figure 6-2: Subroutine for releasing a cylinder of the disk.

- (i) Retrieve the new relations to be created one by one by giving appropriate retrieval commands with output going on the disk (in a prefixed area).
 - (ii) Build these new relations by using BUILDR routine of (Ref. 5) by giving the specification of the new relations and reading the data from the disk.
 - (iii) Remove the older relations which have become redundant as a result of the creation of these new relations.
2. Reorganizing the data base by reorganising the relations such that all data from their overflow area is brought back into regular area and overflow area is cleared. This can be done for each relation to be reorganised as follows:
- (i) Retrieve the relation with output going to disk by giving appropriate retrieval command.
 - (ii) Remove this routine from the data base using REMOVE routine.
 - (iii) Build this relation using BUILDR routine by giving the specification of this relation and reading the data from the disk.

Hence using above steps the DBA can reorganise the data base in any manner he likes.

REFERENCES

1. Agarwala, S., "Updating of a Relational Data Base System on TDC-316", M.Tech. Thesis, Computer Science Programme, Indian Institute of Technology, Kanpur, July 1978.
2. Chamberlin, D.D., "Relational Data-Base Management Systems", ACM Computing Surveys, March 1976, Vol. 8, No. 1, pp. 43-66.
3. Codd, E.F., "A Relational Model of Data for Large Shared Data Banks", Comm. of ACM, Vol. 13, No. 6, June 1970, pp. 370-397.
4. Date, C.J., "An Introduction to Data Base Systems", Addison-Wesley Publishing Company, 1975.
5. Ghonekar, D.K., "An Implementation of a Relational Data Base Model", M.Tech. Thesis, Computer Science Program, Indian Institute of Technology, Kanpur, July 1977.
6. Martin, J., "Principles of Data Base-Management", Prentice-Hall of India Pvt. Ltd., 1977.
7. Sat Pal, "An Information Retrieval System for a Relational Data Base", M.Tech. Thesis, Computer Science Programme, Indian Institute of Technology, Kanpur, July 1978.
8. Sibley, E.H., Fry, James, P., "Evaluation of Data-Base Management Systems", ACM Computing Survey, March 1976, Vol. 8, No. 1, pp. 7-72.
9. Wiederhold, G., "Data Base Design", McGraw-Hill Book Company, 1977.

Appendix 1

INITIAL LOADING OF THE SYSTEM

We have seen that in our system all the programs and security tables reside on the disk. Therefore to load the system initially i.e., to generate the system, the following steps must be followed.

1. Load the "initial loading program" from paper tape into core.

Execute it with starting address = 130000. This will put the routines WORBLE, DVORBL, VORBLE, GIVSEC on the preassigned sectors of the disk and will also initialise the security codes' tables so as to contain only one security code (that of DBA), with its value being 0291234512345 and also DBA's identification number being 000000000000 for values shown in the listings.

2. Load the paper tape for "program to load other programs" as shown in the listings into core.
3. Load paper tapes of all security programs into core except the START program (which is incorporated under KIM) with address ranges as below:

1. PRINTS & TERMNT	70000 to 76000
2. DBA, ALOCTS, RELSEC	76000 to 100240
3. Utility Routines	100400 to 102200
4. INTLIZE	102400 to 104000
5. MODIFY	104100 to 107500

4. Load the paper tapes of REMOVE and PUT routines into core with address range as below:

1. PUT	66000 to 66200
2. REMOVE	66400 to 67700

5. Load the programs developed for BUILDING of a relation developed as part of (Ref. 5) into core with address range as below:

1. SCANNER	27600 to 46400
2. STATIC Routine 1	46400 to 50430
3. BUILD	50440 to 54340
4. INDATA	54400 to 56230
5. STATIC Routine 2	56300 to 57700
6. MAIN	60000 to 65720
6. Execute the "program to load other programs" starting at address 20000. Execution of this program puts all the security programs and programs to build and remove a relation etc., from core into areas of the disk about which information is contained in "Initial loading program."
7. Load the retrieval programs developed as part of (Ref. 7) into core.
8. Load the paper tape of "program to load other programs" into core with X containing the value of symbol NR and Y containing the value of Symbol NR1 of Initial Loading Program.
9. Execute the "program to load other programs" again starting from 20000. This will load all retrieval programs also into preassigned areas of disk.
10. Repeat Steps 7, 8 and 9 for update programs developed as part of (Ref. 1) with X containing value of symbol NU and Y containing value of symbol NU1 of Initial Loading Programs. This will put update programs also into prefixed areas of the disk.

Now the START program incorporated under KDM with code "DB" shall call appropriate programs from the disk depending upon what the user wants to do, as detailed in Appendix 2.

Appendix 2

UNRSERS' MANUAL

To use the system, the following steps must be followed:

Step 1: Press "DB". System prints a message asking the user to specify the first security code.

Step 2: User gives his 13-digit First Security Code as input from the teletype and then line feed. If user is DBA (i.e., the first security code given as input is that of the DBA) then go to Step 5 else go to next step.

Step 3: System asks the user to give his next security code (authority augmenting). User either gives 13-digit security code and then gives "Lf" or gives "N" "Lf". In the former case, the authority vector is enhanced corresponding to the authority augmenting vector associated with that security code and control goes back to Step 3. If input is "N" "Lf", go to next step.

Step 4: System asks the user to specify the job -

- (i) If user gives "R" "Lf" as input from the teletype, system loads Retrieval Programs plus all other utility routines from disk into core and transfers control to retrieval programs. Then user gives retrieval query as detailed in Users' Manual for Retrieval (Ref. 7). After completion of retrieval task control goes back to Step 4.

- (ii) If user gives "U" "Lf" as input from the teletype system loads update programs plus all other/utility routines from disk into core and transfers control to update programs. Then user gives update query as detailed in Users' Manual for Updating (Ref. 1). After completion of update task control goes back to Step 4.
- (iii) If user gives "F" "Lf" then job is finished and system gives "good-bye".
- (iv) If input is none of the above, error message is given and control goes back to Step 4.

Step 5: The DBA is asked to supply the identification number which is a 12-digit number. DBA gives 12-digit number followed by "Lf" or followed by "Y" "Lf". If identification number is correct, then in latter case, DBA is asked to supply new identification number which replaces the old identification number and control goes to next step. In former case control goes directly to next Step.

Step 6: DBA is asked to specify the job. Then DBA specifies one of the jobs as shown on Page 26 of Figure 2-11. If job is "TERMNT", control goes to Step 8 else control goes to Step 7.

Step 7: If job specified is "CCCCCC", then data base is cleared and control goes back to Step 6. If job specified is "INTLZE", "MODIFY" or "REMOVE", DBA follows the procedure mentioned in Chapters 3, 4 and 6 respectively. If job specified is "RELBLD", DBA follows the instructions given in the Users' Manual of (Ref. 5). After completion of these tasks control goes back to Step 6. If job is "PRINTS" DBA follows steps given in Chapter 5, and control goes to next step.

Step 8: All tables from core are stored back on the disk. Job is finished and system gives "GOOD-BYE".

```

*****
*****
: "PROGRAM TO LOAD OTHER PROGRAMS" IS USED TO LOAD SECURITY AND *
: OTHER PROGRAMS USED BY THE DRA INTO PRE ASSIGNED AREAS OF THE *
: DISK. TO LOAD RETRIEVAL PROGRAMS, REPLACE X=ND BY X=NR & Y=ND1 BY *
: Y=NR1. TO LOAD UPDATE PROGRAMS, REPLACE X=ND BY X=NU & Y=ND1 BY *
: Y=NU1. PROCEDURE FOR LOADING CAN BE SEEN FROM APPENDIX1 *
:
:      =      20000
:      R1=      %1
:      R2=      %2
:      R3=      %3
:      R4=      %4
:      NR=      26146
:      ND1=      26150
:      NR=      26152
:      NR1=      26154
:      NU=      26156
:      NU1=      26160
:      X=      ND
:      Y=      ND1
:      ISR      X,R2
:      ISR      Y,R3
LD1:  ISR      (R3)+,R4
:      ISR      (R3)+,STR8
:      ISR      (R3)+,CYL8
LD2:  ISR      (R3)+,SC18
:      JMS      R4,WRITE
CYL8: WORD      0
SC18:  WORD      0
STR8:  WORD      0
:      ADD      #400,STR8
:      DEC      R4
:      BRGT    LD2
:      DEC      R2
:      BRGT    LD1
:      STOP
WRITE:  ISR      #200.##177456
:      BRZS    WRITE
:      ISR      (R4)+,##177452
:      ISR      (R4)+,##177444
:      ISR      (R4)+,##177460
:      ISR      #-128.##177462
:      ISR      #4113.##177454
:      WAIT
SENSE:  ISR      #1.##177456
:      BRZC    SENSE
:      ISR      #40000.##177456
:      BRZS    ERR.11
:      RTS      R4
ERR.11: STOP
:      END

```

```

*****
*****
: INITIAL LOADING PROGRAM PUTS THE ROUTINES VORBLE,DVORGL,GIVSEC *
: AND MO-FILE ON PREASSIGNED AREAS OF THE DISK. IT ALSO INITIALISES *
: VALUES OF VARIOUS VARIABLES AND TABLES AND PUTS THEM ON THE DISK *
: IT'S DATA PORTION CONTAINS ADDRESSES OF THE DISK WHERE PROGRAMS *
: NEEDED BY THE DDA, RETRIEVAL PROGRAMS AND UPDATE PROGRAMS ARE *
: STORED. *
*****
*****
: = 130000
: R1= X1
: R2= X2
: R3= X3
: R4= X4
: R5= X5
: R6= X6
: R7= X7
: MO= 177736
: JMS R4,WRITE
: WORD 200..293..24400
: JMS R4,WRITE
: WORD 200..294..25000
: JMS R4,WRITE
: WORD 200..295..25400
: JMS R4,WRITE
: WORD 200..296..26000
: JMS R4,WRITE
: WORD 200..297..26400
: JMS R4,WRITE
: WORD 200..298..27000
: JMS R4,WRITE
: WORD 196..1..27400
: JMS R4,WRITE
: WORD 196..2..30000
: JMS R4,WRITE
: WORD 196..3..30400
: STOP
WRITE: TSH #200.0#177456
: BRZS WRITE
: TSH (R4)+.0#177452
: TSH (R4)+.0#177444
: TSH (R4)+.0#177460
: TSH #-128..0#177462
: TSH #4113.0#177454
: WAIT
SENSE: TSH #1.0#177456
: BRZC SENSE
: TSH #40000.0#177456
: BRZS ERR.11
: RTS R4
ERR.11: STOP
: = 24400
VORBLE: TSH (R7)+.VVV.1
: TSH VVV.1.(R7)+

```

```

CLR          (R7)+
RIS          R7
V7V.1:      WORD 0
            . = 24500
JVV.1:      TSH  (R7)+.DDD.1
            TSH  (R7)+.DDD.2
            TSH  DDD.1.(R7)+
            RIS  R7
LDD.1:      WORD 0
DDD.2:      WORD 0
            . = 25000
GIVS.1:      TSH  R5, -(R1)
            TSH  R4, -(R1)
            TSH  R3, -(R1)
            TSH  R2, -(R1)
            TSH  #2, R3
            TSH  (R7)+.R5
LG.0:       TSH  #5, R2
CLR          MQ
LG.1:       MMY  #10..R4
            BTSH  (R5)+.R4
            ADD  R4, MQ
            DEC  R2
            BRZC  LG.1
            TSH  MQ, (R7)+
            DEC  R3
            BRZC  LG.0
            TSH  (R1)+.R2
            TSH  (R1)+.R3
            TSH  (R1)+.R4
            TSH  (R1)+.R5
            RIS  R7
LG.TT1:     WORD 0
            . = 25400
WONHLE:     TSH  R4, -(R1)
            TSH  R3, -(R1)
            TSH  R2, -(R1)
            TSH  R5, -(R1)
            TSH  R6, -(R1)
            TSH  (R7)+.R5
            TSH  (R7)+.R6
            TSH  #10..R2
            CLR  R3
            CLR  R4
LPWOR:      BTSH  (R5)+.R3
            BTSH  (R6).R4
            ADD  R4, R3
            CMP  R3, #10.
            BRLT LPW1
            SUB  #10..R3
LPW1:       BTSH  R3, (R6)+
            DEC  R2
            BRZC  LPWOR
            TSH  (R1)+.R6
            TSH  (R1)+.R5

```

```

TSH      (R1)+.R2
TSH      (R1)+.R3
TSH      (R1)+.R4
RTL      R7
.=       26000

```

```

*****
: THE FOLLOWING SEGMENT CONTAINS INITIALISING DATA FOR VARIOUS *
: VARIABLES AND INITIAL IDENTIFICATION NO. OF DMA ETC. *
*****

```

```

DNSEC.1: WORD      29.
ISECYL:  WORD      196.
ISECTR:  WORD      1
ASECYL:  WORD      196.
ASECTR:  WORD      2
FLAG1:   WORD      0
IONS:    WORD      14.,0.,14.
IDNU:    BYTE      60,60,60,60,60,60,60,60,60,60,60,60,60,60
ACYL:    WORD      200.,199.,198.,197.,196.
SITMP:   WORD      -1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
          WORD      0,0,0,0,0,0,0,0,0,0,0,-1
AD:       WORD      2
AD1:      WORD      26162
NR:       WORD      1
NR1:      WORD      26602
NU:       WORD      1
NU1:      WORD      27126

```

```

*****
: THE FOLLOWING SEGMENT CONTAINS INFORMATION ABOUT THE ADDRESSES *
: WHERE PROGRAMS CALLED BY THE DMA GO ON THE DISK *
*****

```

```

WORD      100.,27400,201.
WORD      1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
WORD      33.,34.,35.,36.,37.,38.,39.,40.,41.,42.
WORD      65.,66.,67.,68.,69.,70.,71.,72.,73.,74.
WORD      97.,98.,99.,100.,101.,102.,103.,104.,105.,106.
WORD      129.,130.,131.,132.,133.,134.,135.,136.,137.,138.
WORD      161.,162.,163.,164.,165.,166.,167.,168.,169.,170.
WORD      193.,194.,195.,196.,197.,198.,199.,200.,201.,202.
WORD      225.,226.,227.,228.,229.,230.,231.,232.,233.,234.
WORD      257.,258.,259.,260.,261.,262.,263.,264.,265.,266.
WORD      289.,290.,291.,292.,293.,294.,295.,296.,297.,298.
WORD      10.,11400,202.
WORD      1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
WORD      33.,34.,35.,36.,37.,38.,39.,40.,41.,42.
WORD      65.,66.,67.,68.,69.,70.,71.,72.,73.,74.

```

```

*****
: THE FOLLOWING SEGMENT CONTAINS INFORMATION ABOUT ADDRESSES *
: WHERE RETRIEVAL PROGRAMS ARE STORED ON THE DISK *
*****

```

```

RR:       WORD      70.,54000,202.
          WORD      97.,98.,99.,100.,101.,102.,103.,104.,105.,106.
          WORD      129.,130.,131.,132.,133.,134.,135.,136.,137.,138.
          WORD      161.,162.,163.,164.,165.,166.,167.,168.,169.,170.
          WORD      193.,194.,195.,196.,197.,198.,199.,200.,201.,202.
          WORD      225.,226.,227.,228.,229.,230.,231.,232.,233.,234.

```



```

WORD      257..258..259..260..261..262..263..264..265..266.
WORD      289..290..291..292..293..294..295..296..297..298.

```

```

;*****
; THE FOLLOWING SEGMENT CONTAINS INFORMATION ABOUT AREAS OF DISK *
; WHERE UTILITY ROUTINES ( SCANNER AND IT'S ROUTINES ) AND OTHER *
; TABLES, WHICH NEED TO BE LOADED WITH BOTH RETRIEVAL AND UPDATE *
; PROGRAMS ARE STORED. *
;*****

```

```

WORD      30..27400..201.
WORD      1..2..3..4..5..6..7..8..9..10.
WORD      33..34..35..36..37..38..39..40..41..42.
WORD      65..66..67..68..69..70..71..72..73..74.

```

```

;*****
; THE FOLLOWING SEGMENT CONTAINS INFORMATION ABOUT AREAS OF DISK *
; WHERE UPDATE PROGRAMS ARE STORED *
;*****

```

```

UU: WORD      70..54000..195.
WORD      97..98..99..100..101..102..103..104..105..106.
WORD      129..130..131..132..133..134..135..136..137..138.
WORD      161..162..163..164..165..166..167..168..169..170.
WORD      193..194..195..196..197..198..199..200..201..202.
WORD      225..226..227..228..229..230..231..232..233..234.
WORD      257..258..259..260..261..262..263..264..265..266.
WORD      289..290..291..292..293..294..295..296..297..298.

```

```

;*****
; INITIALISED FIRST SECURITY CODES' TABLE DIRECTORY ( FSECDC ) *
;*****

```

```

      = 27400
WORD      196..3.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0

```

```

;*****
; INITIALISED AUGMENTING SECURITY CODES' TABLE DIRECTORY (ASECDC) *
;*****

```

```

      = 30000
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0
WORD      -1.0.0.0.0.0.0

```

[illegible]

```

*****
:  INITIALISED SECTOR OF FIRST SECURITY CODES' TABLE RESULTING IN
:  SECURITY CODES 0271234512345 AND 0291234512345
*****

```

[illegible]

[illegible]

```

*****
*****
:   START PROGRAM WHICH IS INCORPORATED UNDER KCM WITH CODE 'D-' *
:   ( FOR STARTING EXECUTION FROM ADDRESS 20014 WHICH WILL PRINT *
:   THE SECURITY CODES AS THEY ARE GIVEN FROM THE TELETYPE ) AND *
:   WITH CODE "RD" (FOR STARTING EXECUTION FROM ADDRESS 20000 WHICH *
:   WILL SUPPRESS PRINTING OF SECURITY CODES AS THEY ARE INPUTTED *
:   FROM THE TELETYPE) . *
*****

```

```

:*****
:
:   =          20000
:   K00=        0
:   TTY=        1
:   R1=         %1
:   R2=         %2
:   R3=         %3
:   R4=         %4
:   R5=         %5
:   R6=         %6
:   R7=         %7
:   MC=        17773b
:   VORBLE=     24400
:   DVORBL=    24600
:   GIVSEC=     25000
:   WORBLE=     25400
:   DRSC.1=    26000
:   ISFCYL=     26002
:   ISFCTR=     26004
:   ASFCYL=     26006
:   ASECTR=     26010
:   TSN        #200.AA
:   TSH        #200.SIN.0+2
:*****

```

```

*****
:   THE FOLLOWING PROGRAM SEGMENT READS ROUTINES VORBLE,DVORBL, *
:   GIVSEC,WORBLE AND OTHER INITIAL INFORMATION FROM DISK INTO CORE. *
:*****

```

```

ST.1:   TSN        #137740.R1
:       JMS        R4,READ
:       WORD       200.,.293.,.24400
:       JMS        R4,READ
:       WORD       200.,.294.,.25000
:       JMS        R4,READ
:       WORD       200.,.295.,.25400
:       JMS        R4,READ
:       WORD       200.,.296.,.26000
:       JMS        R4,READ
:       WORD       200.,.297.,.26400
:       JMS        R4,READ
:       WORD       200.,.298.,.27000

```

```

*****
:   THE FOLLOWING SEGMENT ASKS FOR FIRST SECURITY CODE INPUT. IF *
:   FOUND VALID, IT JUMPS TO DBA'S ROUTINE IF SECURITY CODE IS THAT *
:   OF THE DBA. OTHERWISE IT JUMPS TO THE SEGMENT WHICH ASKS FOR *
:   AUTHORITY AUGMENTING SECURITY CODE INPUT. *
:*****

```

RESET

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

WRITE

WAITR

SREAD

WAITR

TSR

TSR

JMS

WORD

WORD

OPD

CLR

RTSR

SUB

P

RTSR

SUB

ADD

MPY

RTSR

SUB

ADD

TSR

TSR

BRIE

CMP

BRT

DEC

CLR

CMP

BRT

INC

SUB

BRT

MPY

TSR

TSR

TSR

CLR

RTSR

SUB

RTSR

DEC

BRT

TSR

TTY, SMC.21

TTY, W21

KAD, SIN.0

KAD, SH2

ISECYL, CYL0

ISECTR, SCT0

R4, READ

0

0

SECSP1

MQ

SIN.0+6, MQ

#60, MQ

#10..R2

SIN.0+7, R2

#60, R2

R2, MQ

#10..R2

SIN.0+8..R2

#60, R2

MQ, R2

R2, R6

R6

ST.4

R6, #91B.

ST.4

R2

MQ

R2, #51.

ST.1.4

MQ

#51..R2

ST.1.3

#14..R3

MQ, SEC.01

MQ, R3

#10..R5

R4

SIN.0+B.(R5), R4

#60, R4

R4, SIN.0+B.(R5)

R5

ST.1.1

SECSP1(1.3)

W21:

S22:

CYL0:

SCT0:

ST.1.3:

ST.1.4:

ST.1.1:

```

      RRGF      ST.2
      JMB      ST.4
      TS      SECSP1(R3),CYL1
      TS      SECSP1+2(R3),SCT1
      JMS      R4,READ
      WORD      0
      WORD      0
      SECSP2
      TS      R2,M0
      MRY      #5,R2
      TS      R0,SFC.02
      TS      M0,R2
      TS      SECSP2+4(R2),BYTE1
      RTGT      BYTE1
      R7S      ST.3
      JMB      ST.4
      TS      #SECSP1+4,SWOR01
      ADD      R3,SWOR01
      TS      #SIN.0+9.,SWOR02
      JMS      R7,WORBLE
      SWOR01:   WORD      0
      SWOR02:   WORD      0
      TS      SWOR02,SWOR03
      JMS      R7,GIVSFC
      SWOR03:   WORD      0
      SWOR04:   =      .+4
      RCOMP     SWOR04,SECSP2(R2)
      RR7C      ST.4
      RCOMP     SWOR04+1,SECSP2+1(R2)
      RR7C      ST.4
      RCOMP     SWOR04+2,SECSP2+2(R2)
      RR7C      ST.4
      RCOMP     SWOR04+3,SECSP2+3(R2)
      RR7C      ST.4
      CMP       R6,DRSC.1
      RR7C      ST.5
      JMP       DBALD
      ST.4:     SRRITE   TTY,SME.22
      W22:      WAITR    TTY,W22
      JMP       ST.1

```

```

;*****
; THE FOLLOWING SEGMENT BUILDS THE USER'S AUTHORITY
; AUGMENTING SECURITY CODES GIVEN BY THE USER. IF AN
; IS GIVEN AS INPUT, THEN THE FIRST SECURITY CODE OF
; CANCELLED
;*****

```

```

MMMMMMMMMMMMMM
CTOR FROM *
WRONG CODE *
HE USER IS *
*
*****

```

```

ST.5:      TSR      ASECYL,CYL2
           TSR      ASECTR,SCT2
           JMS      R4,READ
CYL2:      WORD      0
SCT2:      WORD      0
           WORD      SECSP1
           TSR      #255,R4
ST.5.1:    BCLR     SECSP3(R4)
           DFC      R4

```

ST.5.2:	BRGE	ST.5.1
W23:	SWRITE	TTY,SME.23
	WAITR	TTY,W23
SR3:	SREAD	KBD,SIN.0
	WAITR	KBD,SR3
	RCMP	SIN.0+6.#'N
	BR7C	ST.5.3
	CLR	SIN.0+2
ST.5.3:	JMP	NDBALD
	CLR	MQ
	BTSR	SIN.0+6,MQ
	SUB	#60,MQ
	MPY	#10.,R2
	BTSR	SIN.0+7,R2
	SUB	#60,R2
	ADD	R2,MQ
	MPY	#10.,R2
	BTSR	SIN.0+8.,R2
	SUB	#60,R2
	ADD	MQ,R2
	CLR	MQ
	TST	R2
	BRI F	ST.9
	CMP	R2,#450.
	BRGT	ST.9
ST.5.6:	DEC	R2
	CMP	R2,#25.
	BRLT	ST.5.7
	INC	MQ
	SUB	#25.,R2
ST.5.7:	HRN	ST.5.6
	MPY	#14.,R3
	TSR	MQ,R3
	CLR	R4
ST.5.4:	TSR	#10.,R5
	BTSR	SIN.0+8.(R5),R4
	SUB	#60,R4
	BTSR	R4,SIN.0+8.(R5)
	DEC	R5
	BR7C	ST.5.4
	TST	SECSP1(R3)
	BRGE	ST.6
ST.6:	JMP	ST.9
	TSR	SECSP1(R3),CYL3
	TSR	SECSP1+2(R3),SCT3
	JMS	R4,READ
CYL3:	WORD	0
SCT3:	WORD	0
	WORD	SECSP2
	TSR	R2,MQ
	MPY	#10.,R2
	TSR	MQ,R2
	TST	SECSP2(R2)
	BRGE	ST.7
	JMP	ST.9

```

ST.7:   TSR      #SECSPI+4,SWOR05
        ADD      R3,SWOR05
        TSR      #SIN.0+9.,SWOR06
        JMS      R7,WORBLE
SWOR05: WORD      0
SWOR06: WORD      0
        TSR      SWOR06,SWOR07
        JMS      R7,GIVSEC
SWOR07: WORD      0
SWOR08: . =      .+4
        RCMP     SWOR08,SECSPI2+6(R2)
        BRZC     ST.9
        RCMP     SWOR08+1,SECSPI2+7.(R2)
        BRZC     ST.9
        RCMP     SWOR08+2,SECSPI2+8.(R2)
        BRZC     ST.9
        RCMP     SWOR08+3,SECSPI2+9.(R2)
        BRZC     ST.9
        TSR      SECSPI2(R2),CYL4
        TSR      SECSPI2+2(R2),DVO.13
        TSR      SECSPI2+4(R2),DVO.14
        JMS      R7,DVORBL
DVO.13: WORD      0
DVO.14: WORD      0
DVO.15: WORD      0
        TSR      DVO.15,SCT4
        JMS      R4,READ
CYL4:   WORD      0
SCT4:   WORD      0
        WORD      SECSPI2
        TSR      #255.,R4
ST.8:   BSTB     SECSPI2(R4),SECSPI3(R4)
        DFC      R4
        BRGF     ST.8
        JMP      ST.5.2

```

```

;*****
; THE FOLLOWING SEGMENT IS USED TO CANCELL FIRST SEC RITY CODE OF *
; THE USER WHO ACTS IN AN UNAUTHORISED MANNER *
;*****

```

```

ST.9:   TSR      ISECYL,CYL5
        TSR      ISECTR,SCT5
        JMS      R4,READ
CYL5:   WORD      0
SCT5:   WORD      0
        WORD      SECSPI1
        TSR      SEC.01,R3
        TSR      SECSPI1(R3),CYL6
        TSR      SECSPI1+2(R3),SCT6
        JMS      R4,READ
CYL6:   WORD      0
SCT6:   WORD      0
        WORD      SECSPI2
        TSR      SEC.02,R2
        BSTB     ERBITS,SECSPI2+4(R2)
        TSR      CYL6,CYL7

```



```
TSR      SCT6,SCT7
JMS      R4,WRITE
CYL7:    WORD      0
SCT7:    WORD      0
          WORD      SECSP2
          SWRITE    TTY,SME,22
W22.1:   WAITR     TTY,W22.1
          STOP
AA:      WORD      0
DB:      BCI       %DB%
SIN.0:   WORD      16.,0,16.
          .+16.
SME.21:  WORD      40.,0,40.
          BYTE      15,12
          BCI       % PLEASE GIVE YOUR FIRST SECURI%
          BCI       %TY CODE %
SME.22:  WORD      28.,0,28.
          BYTE      15,12
          BCI       % SORRY.UNABLE TO SERVE YOU%
SME.23:  WORD      60.,0,60.
          BYTE      15,12
          BCI       % PLEASE GIVE YOUR NEXT SECURITY %
          BCI       %CODE(AUTHORITY AUGMENTING)%
SEC.01:  WORD      0
SEC.02:  WORD      0
BYTE1:   BYTE      0,0
```

```
*****
; THE FOLLOWING SEGMENT CONTAINS ROUTINES FOR READING & WRITING *
; ON THE DISK *
*****
```

```
READ:    TSR      #4107,-(R1)

WRITE:    BRN      RW
          TSR      #4113,-(R1)
RW:       TSR      #200.0#177456
          BR7S     RW
          TSR      (R4).0#177452
          CMP      (R4)+.#195.
          BRIT     ERR.12
          TSR      (R4)+.0#177444
          TSR      (R4)+.0#177460
          TSR      #-128.0#177462
          TSR      (R1)+.0#177454

SENSE:    TSR      #1.0#177456
          BR7C     SENSE
          TSR      #40000.0#177456
          BR7S     ERR.11
          RTS      R4

ERR.12:   STOP
ERR.11:   WORD      12.
ERR.11:   STOP
SECSP1:   .+256.
SECSP2:   .+256.
SECSP3:   .+256.
          STDBA=    75762
```

```

VDBAST= 54000
AC= 26146
AD1= 26150
AR= 26152
BR1= 26154
BU= 26156
NU1= 26160

```

```

*****
; THIS SEGMENT LOADS SECURITY,BUILD AND REMOVE PROGR S ETC. *
; CEEDED BY THE DBA AND THEN JUMPS TO DBA'S ROUTINE *
*****

```

```

DBALD:  TSR      ND,R2
        TSR      ND1,R3
LD1:    TSR      (R3)+,R4
        TSR      (R3)+,STR8
        TSR      (R3)+,CYL8
LD2:    TSR      (R3)+,SCT8
        JMS      R4,READ
CYL8:   WORD     0
SCT8:   WORD     0
STR8:   WORD     0
        ADD      #400,STR8
        DEC      R4
        BRGT     LD2
        DEC      R2
        BRGT     LD1
        JMP      STDBA

```

```

*****
; THIS SEGMENT LOADS PROGRAMS FOR RETRIEVAL OR UPDAT DEPENDING *
; UPON USER'S INTEREST & JUMPS TO THE CORRESPONDING UTINE *
*****

```

```

NDBALD:  SWRITE   TTY,SME.30
WNR1:    WAITR    TTY,WNR1
        SREAD     KBD,SIN,0
WNR2:    WAITR    KBD,WNR2
        RCOMP     SIN,0+6,#'R
        BR7S      RETRVL
        RCOMP     SIN,0+6,#'U
        BR7S      UPDATE
        RCOMP     SIN,0+6,#'F
        BR7C      NDBALD
        SWRITE    TTY,SME.31
WNR3:    WAITR    TTY,WNR3
        STOP
        NOP
RETRVL:  TSR      NR,R2
        TSR      NR1,R3
        JMP      LD3
UPDATE:  TSR      NU,R2
        TSR      NU1,R3
        SUB      #66.,R3
LD3:     INC      R2
LD3.1:   TSR      (R3)+,R4
        TSR      (R3)+,STR9
        TSR      (R3)+,CYL9

```

```

LD4:      TSH      (R3)+.SCT9
          JMS      R4,HEAD
CYL9:     WORD     0
SCT9:     WORD     0
STR9:     WORD     0
          ADD      #400,STR9
          DEC      R4
          BRGT    LD4
          DEC      R2
          BRGT    LD3.1
          JMP      NDBAST
SME.30:   WORD     18,,0,18.
          BYTE     15,12
          BCI      % SPECIFY THE JOB%
SME.31:   WORD     12,,0,12.
          BYTE     15,12
          BCI      % GOOD BYE %

```

```

;*****
;   THE FOLLOWING ROUTINE IS USED TO CHECK WHETHER USE S REQUEST
;   FALLS WITHIN HIS AUTHORITY. IF NOT, HIS FIRST SECU TY CODE IS
;   CANCELLED
;*****

```

```

          L=        24200
CHKSEC:   TSH      R3,=(R1)
          TSH      R2,=(R1)
          CLH      R2
          CLH      R3
SL.1:     RCIB     SECSP3(R2),SECSP1(R2)
          BTSR     SECSP1(R2),R3
          TST      R3
          BRZC     SL.2
          INC      R2
          CMP      R2,#256.
          BRIT     SL.1
          TSH      (R1)+,R2
          TSH      (R1)+,R3
          RTS      R7
SL.2:     TSH      #3,ERBITS
          JMP      ST.9
          END

```

```

*****
*****
; PRINTS PROGRAM WHICH IS USED TO PRINT ALL SECURITY TABLES & ALSO*
; AUTHORITY VECTORS AS REQUESTED BY THE DDA 1
*****
*****

```

```

      . = 70000
      R1 = X1
      R2 = X2
      R3 = X3
      R4 = X4
      R5 = X5
      R6 = X6
      R7 = X7
      MO = 17773b
      KBD = 0
      TTY = 1
      PRN = 6
      ISFCYL = 26002
      ISFCTR = 26004
      ASFCYL = 26006
      ASFCTR = 26010
      DVORHL = 24600
      SFCSP1 = 22170
      SFCSP2 = 22570
      SFCSP3 = 23170
      READ = 022060
      WRITE = 022066
      EXIT = 76214
PRINTS:  TSR      R7, -(R1)
          TSR      R6, -(R1)
          TSR      R5, -(R1)
          TSR      R4, -(R1)
          TSR      R3, -(R1)
          TSR      R2, -(R1)
          TSR      R1,SSAVEP

```

```

      RESET
S.71:  SWRITE     TTY,SME.14
W14:   WAITR      TTY,W14
      SREAD      KBD,SIN.4
WHP1:  WAITR      KBD,WHP1
      CMP        SIN.4+6, #*PR
      BR7C       S.72
      TSR        PRNTR,INPUTP
      BRN        S.73
S.72:  CMP        SIN.4+6, #*TT
      BR7C       S.71
      TSR        TELTYP,INPUTP
      TSR        #5015.TBF1
      TSR        #5015.TBF2
      TSR        #5015.TBF3
      TSR        #5015.TBF4
      TSR        #5015.TBF5
      TSR        #5015.TBF6
      TSR        #5015.TBF7

```

	TSR	#5015.T0F8
	TSR	#5015.T0F9
S.73:	TSR	#5015.T0F10
	INIT	PRN,INPUT
W19:	SWRITE	TTY,SME.15
	WAITR	TTY,W15
	SRPAD	KBD,SIN.4
	WNP2:	WAITR KBD,WNP2
	RCMP	SIN.4+0.M'Y
	BRZS	S.74
	JMP	S.HB
S.74:	TSR	ISECYL,CYL.41
	TSR	ISECTR,SCT.41
	JMS	R4,READ
CYL.41:	WORD	0
SCT.41:	WORD	0
	WORD	SECSP1
	CLN	R6
	SWRITE	PRN,PBF1
WP1:	WAITR	PRN,WP1
S.P.11:	SWRITE	PRN,PBF10
WP2:	WAITR	PRN,WP2
	SWRITE	PRN,PBF2
WP3:	WAITR	PRN,WP3
	SWRITE	PRN,PBF10
WP4:	WAITR	PRN,WP4
	TSR	#-14..R2
	CLR	PCOUNT
	CLN	R3
S.P.11:	TSR	#20040.PBF3+36.(R3)
	ADD	#2,R3
	CMP	R3,#70.
	BRZC	S.P.11
S.75:	INC	PCOUNT
	CMP	PCOUNT,#19.
	BRZS	S.76
	ADD	#14..R2
	TST	SECSP1(R2)
	BRLT	S.75
	JMS	R7,POWR
	WORD	2
	WORD	PCOUNT
	WORD	PBF3+47.
	TSR	SECSP1(R2).PTMP
	JMS	R7,POWR
	WORD	3
	WORD	PTMP
	WORD	PBF3+58.
	TSR	SECSP1+2(R2).PTMP
	JMS	R7,POWROC
	WORD	PTMP
	WORD	PBF3+70.
	TSR	#SECSP1+4.PTEMP
	ADD	R2,PTMP
	JMS	R7,COPC

	WORD	10.
	WORD	PTEMP
	WORD	PBF3+83.
WP5:	SWRITE	PRN,PBF3
	WAITR	PRN,WP5
S.76:	BRN	S.75
	TST	R6
	BR7S	S.P.2
	JMP	S.90
S.P.2:	TSR	#-14.,R2
	TSR	#19.,R3
WP6:	SWRITE	PRN,PBF4
	WAITR	PRN,WP6
WP7:	SWRITE	PRN,PBF10
	WAITR	PRN,WP7
WP8:	SWRITE	PRN,PBF5
	WAITR	PRN,WP8
WP9:	SWRITE	PRN,PBF10
	WAITR	PRN,WP9
	CLR	R4
S.P.12:	TSR	#20040,PBF3+36.(R4)
	ADD	#2,R4
	CMP	R4,#70.
	BR7C	S.P.12
	CLR	R5
S.77:	ADD	#14.,R2
	DEC	R3
	BR2C	S.P.15
	JMP	S.85
S.P.15:	TST	SECSP1(R2)
	BRLT	S.77
	TSR	SECSP1(R2),CYL.42
	TSR	SECSP1+2(R2),SCT.42
	JMS	R4,READ
CYL.42:	WORD	0
SCT.42:	WORD	0
	WORD	SECSP2
	TSR	#18.,PCOUNT
	SUB	R3,PCOUNT
	TSR	PCOUNT,H0
	MPY	#51.,PCOUNT
	TSR	M0,PCOUNT
	TSR	#-5,R4
S.78:	INC	PCOUNT
	ADD	#5,R4
	CMP	R4,#250.
	BRGT	S.77
	BTSR	SECSP2+4(R4),BYT4
	BCIB	BYT1,BYT4
	BCMP	BYT4,#48.
	BR7S	S.78
	BTSR	SECSP2+4(R4),BYT4
	BCIB	BYT2,BYT4
	BCMP	BYT4,#12.
	BR7C	S.79

	TSR	ILLGL6,PBF3+46.(R5)
	TSR	ILLGL6+2,PBF3+48.(R5)
	TSR	ILLGL6+4,PBF3+50.(R5)
	BRZ	S.P.13
S.77:	BTSR	SECSP2+4(R2),BYT4
	BCD	BYT3,BYT4
	BCMP	BYT4,#3
	BRZC	S.80
	TSR	MORASK,PBF3+46.(R5)
	TSR	MORASK+2,PBF3+48.(R5)
	TSR	MORASK+4,PBF3+50.(R5)
	BRZ	S.P.13
S.80:	TSR	#20040,PBF3+46.(R5)
	TSR	#20040,PBF3+48.(R5)
	TSR	#20040,PBF3+50.(R5)
S.P.13:	BTSR	SECSP2(R4),PTEMP
	BTSR	SECSP2+1(R4),PTEMP+1
	JMS	R7,POWR
	WORD	5
	WORD	PTEMP
S.81:	WORD	PBF3+57.
	BTSR	SECSP2+2(R4),PTEMP
	BTSR	SECSP2+3(R4),PTEMP+1
	JMS	R7,POWR
	WORD	5
	WORD	PTEMP
S.82:	WORD	PBF3+62.
	JMS	R7,POWR
	WORD	3
	WORD	PCOUNT
S.83:	WORD	PBF3+54.
	INC	R6
	CMP	R6,#1
	BRZC	S.84
	ADD	#29..S.81
	ADD	#29..S.82
	ADD	#29..S.83
	TSR	#28..R5
	BRN	S.78
S.84:	SUB	#29..S.81
	SUB	#29..S.82
	SUB	#29..S.83
	CLR	R5
	CLR	R6
	SWRITE	PRN,PBF3
WP10:	WAITR	PRN,WP10
	BRN	S.78
S.85:	IST	R6
	BRZS	S.88
	TSR	#25..R2
S.86:	DEC	R2
	BRZS	S.87
	BTSR	#40,PBF3+73.(R2)
	BRN	S.86
S.87:	SWRITE	PRN,PBF3

WP11:	WAITR	PRN,WP11
S.88:	SWRITE	TTY,SME.16
W16:	WAITR	TTY,W16
	SKFAD	KRD,SIN.4
WRP3:	WAITR	KRD,WRP3
	BCMP	SIN.4+6, #'Y
	BRZS	S.89
	JMP	ENDPRI
S.89:	TSR	#1,R6
	TSR	ASECYL,CYL.43
	TSR	ASECTH,SCT.43
	JMS	R4,READ
CYL.43:	WORD	0
SCT.43:	WORD	0
	WORD	SECSP1
	SWRITE	PRN,PBF6
WP12:	WAITR	PRN,WP12
	JMP	S.P.1
S.90:	SWRITE	TTY,SME.17
W17:	WAITR	TTY,W17
	SKFAD	KRD,SIN.4
WRP4:	WAITR	KRD,WRP4
	CLR	R6
	BTSR	SIN.4+6,R6
	SUB	#60,R6
	TSR	R6,MQ
	MPY	#10,R6
	BTSR	SIN.4+7,R6
	SUB	#60,R6
	ADD	R6,MQ
	MPY	#10,R6
	BTSR	SIN.4+8,R6
	SUB	#60,R6
	ADD	MQ,R6
	TST	R6
	BRLT	S.90
	CMP	R6,#256
	BRLF	S.91
	TSR	#256,R6
S.91:	SWRITE	PRN,PBF7
WP121:	WAITR	PRN,WP121
	SWRITE	PRN,PBF10
WP13:	WAITR	PRN,WP13
	TSR	#-14,R2
	TSR	#19,R3
	CLR	R4
S.P.14:	TSR	#20040,PBF3+36,(R4)
	ADD	#2,R4
	CMP	R4,#70
	BRZC	S.P.14
S.92:	ADD	#14,R2
	DEC	R3
	BRZS	ENDPRI
	TST	SECSP1(R2)
	BRLT	S.92

	TSR	SECSP1(R2),CYL.44
	TSR	SECSP1+2(R2),SCT.44
	JMS	R4,READ
CYL.44:	WORD	0
SCT.44:	WORD	0
	WORD	SECSP2
	TSR	*18.,PCOUNT
	SDH	R3,PCOUNT
	TSR	PCOUNT,MU
	MPY	#25.,PCOUNT
	TSR	MU,PCOUNT
	TSR	#-10.,R4
	TSR	#26.,R2
S.93:	DEC	R5
	GR2S	S.92
	ADD	#10.,R4
	INC	PCOUNT
	TSR	SECSP2(R4)
	BLT	S.93
	TSR	SECSP2(R4),CYL.45
	TSR	SECSP2+2(R4),DVO.10
	TSR	SECSP2+4(R4),DVO.11
	JMS	R7,DVORBL
DVO.10:	WORD	0
DVO.11:	WORD	0
DVO.12:	WORD	0
	TSR	DVO.12,SCT.45
	JMS	R4,READ
CYL.45:	WORD	0
SCT.45:	WORD	0
	WORD	SECSP3
	JMS	R7,POWR
	WORD	3
	WORD	CYL.45
	WORD	PBF8+79.
	JMS	R7,POWRDC
	WORD	SCT.45
	WORD	PBF8+94.
	TSR	SECSP2+6(R4),PTMP
	JMS	R7,POWR
	WORD	5
	WORD	PTMP
	WORD	PBF8+55.
	TSR	SECSP2+8.(R4),PTMP
	JMS	R7,POWR
	WORD	5
	WORD	PTMP
	WORD	PBF8+60.
	TSR	PCOUNT,PTMP
	JMS	R7,POWR
	WORD	3
	WORD	PTMP
	WORD	PBF8+52.
	WRITE	PRN,PBF8
WP14:	WAITR	PRN,WP14

SWRITE

PRN, PRF9

```

WP12:  WAITP  PRN, WP15
        SWRITE PRN, PRF10
WP13:  WAITR  PRN, WP16
        TSN    R6, NUMBPR
        JMN    R7, PR1SEC
AUX1P0: WORD    0
        WORD    SECUP3
        WORD    PRF3+41.
        JMP     S.93
ENDPNI: TSN     SSAVER, R1
        TSN     (R1)+, R2
        TSN     (R1)+, R3
        TSN     (R1)+, R4
        TSN     (R1)+, R5
        TSN     (R1)+, R6
        TSN     (R1)+, R7
        JMP     TERMINT
PRF1:  WORD    100..0.100.
        BCI     %
TBF1:  BCI     % FIRST %
        BCI     %SECURITY CODES' TABLE %
        BCI     %DIRECTORY %
PRF2:  WORD    100..0.100.
        BCI     %
TBF2:  BCI     % S.NO. CYLINDER%
        BCI     % NO. SECTOR NO. ROTATING %
        BCI     %DIGITS %
PRF3:  WORD    100..0.100.
        BCI     %
TBF3:  BCI     %
        BCI     %
        BCI     %
PRF4:  WORD    100..0.100.
        BCI     %
        BYTE    15,12,15,12
TBF4:  BCI     %
        BCI     %FIRST SECURITY CODES' %
        BCI     %TABLE %
PRF5:  WORD    100..0.100.
        BCI     %
TBF5:  BCI     % STATUS SECURITY%
        BCI     % CODE STATUS %
        BCI     %SECURITY CODE %
PRF6:  WORD    100..0.100.
        BCI     %
        BYTE    15,12,15,12
TBF6:  BCI     % AUGMENTING %
        BCI     %SECURITY CODES' TABLE %
        BCI     %DIRECTORY %
PRF7:  WORD    100..0.100.
        BCI     %
        BYTE    15,12,15,12
TBF7:  BCI     % AUGMENTING SECURITY %
        BCI     %CODES' TABLE WITH AUTHORITY %
        BCI     %VECTORS %

```

```

PRF8:      WORD      100., 0., 100.

THF8:      BCI      %
            BCI      % SECURITY CODE=
            BCI      % CYLINDER NO.=
            BCI      % SECTOR NO.=
PRF9:      WORD      100., 0., 100.
            BCI      %
THF9:      BCI      % CORRESPONDING AUTHORITY
            BCI      % VECTOR IS AS BELOW:
            BCI      %
PRF10:     WORD      2, 0, 2
THF10:     WORD      2040
SSAVER:    WORD      0
SIN.4:     WORD      14., 0., 14.
            =
            .+14.
PRNTR:     WORD      10
INPUT:     WORD      0
TELYP:     WORD      2
PTMP:      WORD      0
PCOUNT:    WORD      0
BYT1:      BYTE      -49.
BYT2:      BYTE      -13.
BYT3:      BYTE      -4
BYT4:      BYTE      0.
ILLOLE:    BCI      %ILLOLE%
MORASK:    BCI      %MORASK%
SMF.14:    WORD      64., 0., 64.
            BYTE      15, 12
            BCI      % SPECIFY THE DEVICE WHERE%
            BCI      % SECURITY TABLES ARE TO BE PRINTED %
SME.15:    WORD      56., 0., 56.
            BYTE      15, 12
            BCI      % DO YOU WANT TO PRINT%
            BCI      % FIRST SECURITY CODES' TABLES? %
SME.16:    WORD      60., 0., 60.
            BYTE      15, 12
            BCI      % DO YOU WANT TO PRINT AUG%
            BCI      % MENTING SECURITY CODES' TABLES? %
SME.17:    WORD      54., 0., 54.
            BYTE      15, 12
            BCI      % GIVE NO. OF FIELDS IN AUTHORITY%
            BCI      % VECTORS IN 3 DIGITS%

```

```

;*****
; THIS ROUTINE PRINTS A SECTOR OF AUTHORITY VECTOR & IS CALLED BY *
; THE PRINTS ROUTINE .
;*****

```

```

PRISEC:    TSR      R5, -(R1)
            TSR      R4, -(R1)
            TSR      R3, -(R1)
            TSR      R2, -(R1)
            TSR      (R7)+, R2
            TSR      (R7)+, R3
            CLR      PDL1
            TSR      (R7)+, STPB
PHLO:      TSR      #20, R4
            TSR      STPB, R5

```

```

PRL1:  BTSR      (R3)+,BYT5
        BCLR     BYT5,BYT6
        BCLR     BYT7,BYT5
        BCLR     BYT8,BYT6
        BCOMP    BYT5,#9.
        BRGT     PRL2
        BTSR     BYT5,PDL1
        ADD      #60,PDL1
        BRN      PRL3
PRL2:  BTSR     BYT5,PDL1
        ADD      #67,PDL1
PRL3:  BCLR     BYT5
        LSL      BYT6
        LSL      BYT6
        LSL      BYT6
        LSL      BYT6
        BTSR     PDL1,BYT6
        BCOMP    BYT5,#9.
        BRGT     PRL4
        BTSR     BYT5,PDL1
        ADD      #60,PDL1
        BRN      PRL5
PRL4:  BTSR     BYT5,PDL1
        ADD      #67,PDL1
PRL5:  BTSR     PDL1,(R5)+
        BTSR     BYT6,(R5)+
        BTSR     #40,(R5)+
        DEC      R2
        BRLE     PRL6
        DEC      R4
        BR7C     PRL1
        SWRITE   PRN,PBF3
WP17:  WAITR    PRN,WP17
        BRN      PRL0
PRL6:  DEC      R4
        TSH      R4,M0
        MPY      #3,R4
        TSH      M0,R4
PRL7:  BTSR     #40,(R5)+
        DEC      R4
        BRGT     PRL7
        SWRITE   PRN,PBF3
WP18:  WAITR    PRN,WP18
        TSH      (R1)+,R2
        TSH      (R1)+,R3
        TSH      (R1)+,R4
        TSH      (R1)+,R5
        RTS      R7
PDL1:  WORD     0
BYT8:  BYTE     15.
BYT7:  BYTE     -16.
BYT6:  BYTE     0
BYT5:  BYTE     0
STPB:  WORD     0

```

: COPC,POWROC & POWR ARE A FEW UTILITY ROUTINES CALLED BY PRINTS *
 : PROGRAM & ARE SELF EXPLANATORY. *

```

COPC:      TSH      R5,-(R1)
           TSH      R4,-(R1)
           TSH      R3,-(R1)
           TSH      R2,-(R1)
           CLW      R5
           TSH      (R7)+,R2
           TSH      @ (R7)+,R3
           TSH      (R7)+,R4
LCP1:      BTSH     (R3)+,R5
           ADD      #60,R5
           BTSH     R5,(R4)+
           DEC      R2
           BRZC     LCP1
           TSH      (R1)+,R2
           TSH      (R1)+,R3
           TSH      (R1)+,R4
           TSH      (R1)+,R5
           RTS      R7
POWROC:    TSH      R5,-(R1)
           TSH      R4,-(R1)
           TSH      R3,-(R1)
           TSH      R2,-(R1)
           TSH      @ (R7)+,R2
           TSH      (R7)+,R3
           TSH      R2,R4
           CLW      PW1,R2
           CLW      PW2,R4
           BTSH     #60,(R3)+
           ASH      R2
           ASH      R2
           ASH      R2
           ASH      R2
           ASH      R2
           ADD      #60,R2
           BTSH     R2,(R3)+
           BTSH     #1,,(R3)+
           CMP      R4,#10
           BRZS     PP1
           ADD      #60,R4
           BTSH     #60,(R3)+
           BTSH     R4,(R3)+
           BRN      PP2
PP1:      BTSH     #61,(R3)+
           BTSH     #60,(R3)+
PP2:      TSH      (R1)+,R2
           TSH      (R1)+,R3
           TSH      (R1)+,R4
           TSH      (R1)+,R5
           RTS      R7
PW1:      WORD     -481
PW2:      WORD     -32
POWR:     TSH      R2,-(R1)
  
```

```

TSR      R4,=(R1)
TSR      R6,=(R1)
TSR      (R7)+,R2
TSR      *(R7)+,R4
TSR      (R7)+,XT

```

```

ADD      R2,XT
LPP.1:   TST      R2
HRZS     RTND
DEC      R2
DEC      XT
TSR      R4,MQ
CLR      R4
DIV      #10.,R4
ADD      #60,R4
BTSR     R4,XT
TSR      MQ,R4
BRN      LPP.1

```

```

RTND:    TSR      (R1)+,R6
          TSR      (R1)+,R4
          TSR      (R1)+,R2
          RTS      R7

```

```

XT:      WORD     0

```

```

;*****
;*****
;   TERMNT PROGRAM WHICH PUTS VARIABLE CORE SECURITY DATA BACK ON
;   AND FINISHES THE JOB
;*****
;*****

```

```

TERMNT:  RESET
          JMS      R4,WRITE
          WORD     200.,296.,26000
          SWRITE   1,TMES
WT:      WAITR    1,WT
          STOP
TMES:    WORD     10.,0.10.
          BYTE     15,12
          BCI      %GOOD BYE%
          END

```

```

*****
*****
: THE FOLLOWING IS THE DBA'S PROGRAM WHICH ASKS THE DBA FOR HIS
: IDENTIFICATION & IF VALID, DOES THE JOBS REQUESTED BY HIM
*****
*****

```

```

L= 75762
KBD= 0
TTY= 1
MO= 177736
SM= 5
PON= 6
R1 = %1
R2= %2
R3= %3
R4= %4
R5= %5
R6= %6
R7= %7
TERMNT= 75646
PRINTS= 70000
MODIFY= 104100
INTLZE= 102400
FLAG1= 26012
IONS= 26014
IDNO= 26022
AA= 21576

```

```

RESET
DMA: SWRITE TTY,SMES.1
W1: WAITR TTY,W1
TSR AA,SIN.1+2
SREAD KBD,SIN.1
WH1: WAITR KBD,WR1
S.1: TSR #12..R2
TSR #IDNO,R3
TSR #SIN.1+6,R4
S.2: RCMP (R3)+.(R4)+
BRZC SEHHR
DEC R2
BRZC S.2
RCMP (R4).#Y
BRZC S.3
SWRITE TTY,SMES.4
W2: WAITR TTY,W2
TSR AA,IONS+2
SREAD KBD,IONS
WR2: WAITR KBD,WR2
CLR IONS+2
CLR SIN.1+2
S.3: BTST FLAG1
BRZS S.4
BTSR FLAG1,SMES.3+41.
ADD #30000,SMES.3+40.
SWRITE TTY,SMES.3
W3: WAITR TTY,W3

```

	MOVP	FLAG1.#9.
	BR7S	SENHOR
	MOVM	FLAG1
S.4:	SWRITE	TTY,SMES.5
W4:	WAITR	TTY,W4
	BPA	S.4.2
S.4.1:	SWRITE	TTY,SME.6
W4.1:	WAITR	TTY,W4.1
S.4.2:	SRPAD	KUD,SIN.1
W4.3:	WAITR	KUD,W4.3
	TSR	FNCTN,R2
	TSR	#CUDA.1,R3
L.1:	TSR	#SIN.1+6,R4
	TSR	(R3)+.R6
	TSR	LNGTJ,R5
L.2:	HCOMP	(R6)+.(R4)+
	BR7C	NXT
	DFC	R5
	BR7C	L.2
	ISI	R2
	JMP	BR-2(R2)
NXT:	DFC	R2
	BR7S	S.4
	BRN	L.1
SENHOR:	HINC	FLAG1
	BCMP	FLAG1.#9.
	BRIE	S.5
	BYSR	#9.,FLAG1
S.5:	SWRITE	TTY,SMES.2
W5:	WAITR	TTY,W5
	JMP	TERMNT
SIN.1:	WORD	16.,0,16.
+16.
FNCTN:	WORD	7
LNGTJ:	WORD	6
SMES.1:	WORD	44.,0,44.
	BYTE	15,12
	BCI	% PLEASE GIVE YOUR IDENTIFICATION%
	BCI	% NUMBER %
SMES.2:	WORD	30.,0,30.
	BYTE	15,12
	BCI	% SORRY.UNABLE TO SERVE YOU %
SMES.3:	WORD	42.,0,42.
	BYTE	15,12
	BCI	% ILLEGAL REFERENCE MADE %
	BCI	%EARLIER TIMES%
SMES.4:	WORD	34.,0,34.
	BYTE	15,12
	BCI	% GIVE NEW IDENTIFICATION%
	BCI	% NUMBER %
SMES.5:	WORD	18.,0,18.
	BYTE	15,12
	BCI	% SPECIFY THE JOB%
SME.6:	WORD	30.,0,30.
	BYTE	15,12


```

      HCTI          % JOB OVER, SPECIFY NEXT JOB %
CODA.1: WORD       CD1,CD2,CD3,CD4,CD5,CD6,CD7
CD1:   BCI         %INTLZE%
CD2:   HCTI       %MODIFY%
CD3:   HCTI       %REMOVE%
CD4:   HCTI       %TERMNT%
CD5:   BCI        %PRINTS%
CD6:   HCTI       %CCCCC%
CD7:   BCI        %RELBLD%
BH:    WORD       RELBLD
      WORD        CCCCC
      WORD        PRINTS
      WORD        TERMNT
      WORD        REMOVE
      WORD        MODIFY
      WORD        INTLZE
      RELBLD=     60040
      CCCCC=      56630

```

```

CCCCC= 56630
REMOVE= 66400

```

```

;*****
; THE FOLLOWING ROUTINE IS USED TO ASSIGN A SECTOR 0 SECURITY
; SPACE TO CALLING PROGRAM. CALL IS JMS R7,ALOCTS & T RETURNS
; CYL. NO. & SECTOR & SURFACE NO. TO THE NEXT TWO WO S
;*****

```

```

      . =          77106
      ACYL =       26036
      BITMP =      26050
ALOCTS: TSR        R6,-(R1)
      TSR        R5,-(R1)
      TSR        R4,-(R1)
      TSR        R3,-(R1)
      TSR        R2,-(R1)
      TSR        R1,SSAVE1
      RESET
      JMS        R2,RNDY1
R:     WORD      0
      TSR        R,MQ
      CLR        R
      DIV        #62..R
      ASR        R
      LSL        R
S.11:  TSR        R,R2
      TSR        #62..R3
      CLR        R6
S.12:  TSR        #16..R4
      TSR        #1..R5
S.13:  TSB        R5,BITMP(R2)
      BRZS       GOTONE
      LSL        R5
      INC        R6
      DEC        R4
      BRZC       S.13
      ADD        #2,R2
      CMP        R2,#62.
      BRZC       S.14

```

```

S.14:  CLR      R2
      SUB      #2,R3
      MZC
      WRITE    TTY,SMES.6
W1:    WAITR    TTY,W6
      TSR      EMPTYS,(R7)+
      ADD      #2,R7
      ORN      ES.2
GUTUM:  STX     R5,HIMP(R2)
      ADD      R,R
      ADD      R,R
      ADD      R,R
      ADD      R6,R
      CMP      R,#496.
      BRIT     S.15
S.15:  SUB      #496.,R
      TSR      R,MQ
      CLR      R
      DIV      #100.,R
      LSL      MQ
      TSR      MQ,R2
      TSR      ACYL(R2),(R7)+
      TSR      R,MQ
      CLR      R
      DIV      #10.,R
      MPY      #32.,R5
      ADD      MQ,R
      INC      R
      TSR      R,(R7)+
ES.2:  TSR      SSAVE1,R1
      TSR      (R1)+,R2
      TSR      (R1)+,R3
      TSR      (R1)+,R4
      TSR      (R1)+,R5
      TSR      (R1)+,R6
      RTS      R7
SSAVE1: WORD    0
EMPTYS:  WORD    -1
SMES.6:  WORD    42.,0.42.
      BYTE     15,12
      BCI      % NO SECTOR AVAILABLE IN %
      BCI      %SECURITY SPACE %

```

```

;*****
;  FOLLOWING ROUTINE RELEASES A SECTOR OF DISK SECURITY SPACE WITH *
;  CALL JMS R7,RELSEC & NEXT TWO WORDS CONTAINING THE ADDRESS OF *
;  THE SECTOR TO BE RELEASD .                                     I *
;*****

```

```

RELSEC:  TSR      R6,-(R1)
      TSR      R5,-(R1)
      TSR      R4,-(R1)
      TSR      R3,-(R1)
      TSR      R2,-(R1)
      TSR      R1,SSAVE2
      RESET
      TSR      #0,R2

```

```

S.21:   TSR      (R7)+,CYLSEC
        TSR      (R7)+,SECTRS
        TSR      #0,R3
        CMP      CYLSEC,ACYL(R3)
        BRZS     S.22
        ADD      #2,R3
        INC      R2
        CMP      R2,#5
        BRZS     S.24
S.22:   BRN      S.21
        TSR      SECTRS,M0
        TSR      #0,R3
        DIV      #32,,R3
        DEC      R3
        MPY      #10,,R4
        ADD      M0,R3
        TSR      R2,M0
        MPY      #1001,,R2
        ADD      R3,M0
        CMP      M0,#495.
        BRGT     S.24
        CLR      R3
        DIV      #16,,R3
        INC      R3
        TSR      M0,R4
        ADD      R4,R4
        TSR      #1,SSSSS1
S.23:   DEC      R3
        NOP
        BRZS     S.23.1
        LSL      SSSSS1
S.23.1: BRN      S.23
        CLB      SSSSS1,BITMP(R4)
        NOP
        BRN      ES.3
SSSSS1: WORD      0
S.24:   SWRITE   TTY,SMES.7
W7:     WAITR    TTY,W7
ES.3:   TSR      SSAVE2,R1
        TSR      (R1)+,R2
        TSR      (R1)+,R3
        TSR      (R1)+,R4
        TSR      (R1)+,R5
        TSR      (R1)+,R6
        RTS      R7
SMES.7: WORD      76,,0,76.
        BYTE     15,12
        BCI      X SECTOR TO BE RELEASED OUT OF X
        BCI      X RANGE,NO SECTOR RELEASED%
        BCI      X,PROGRAM CONTINUES X
SSAVE2: WORD      0
CYLSEC: WORD      0
SECTRS: WORD      0

```

```

*****
;   THIS IS A RANDOM NO. GENERATOR FOR THE ENTIRE SYSTEM
*****
RNDY1:   TSH      N1,N3
        ADD      N2,N3
        TSH      N2,N1
        TSH      N3,N2
        TSH      N3,(N2)+
        RTS      R2
N1:      WORD     0
N2:      WORD     1
N3:      WORD     0
        END

```

```

;*****
;*****
;  UTILITY ROUTINES
;*****
;*****

```

```

      = 100400
      = 177736
      = %1
      = %2
      = %3
      = %4
      = %5
      = %6
      = %7
      = 0
      = 1
      = 5
      = 6
      = 100154

```

```

;*****
;  THE FOLLOWING ROUTINE IS USED TO GET A CHARACTER OF INPUT FROM *
;  THE DEVICE WHOSE CODE IS CONTAINED IN INPUT. CALL JMS R7,GET*
;  & CHARACTER RETURNED IS IN VARIABLE CH.
;*****

```

```

GET:   TSR      R4, -(R1)
        TSR      R1, SSAVEG
        RTST     FLAG
        BRZC     S.35
        BTR      #1, FLAG
        CLR      SAVE1
        CLR      COUNT
        BCOMP    INPUT, #1
        BRZC     S.34
        TSR      #2, XTSUB
S.34:   BRN      S.35
S.35:   CLR      XTSUB
        TST      COUNT
        BRZC     S.37
        CLR      SAVE1
S.36:   RESET
        INIT     SN, INPUT
        INIT     PRNT, PRNTR
        SWRITE   TTY, XB
BW:     WAITR    TTY, BW
        SREAD    SN, IN2
WW2:    WAITR    SN, WW2
        SUB      XTSUB, COUNT
        TST      COUNT
        BRLE     S.38
        SWRITE   PRNT, IN2
WW3:    WAITR    PRNT, WW3
S.37:   TSR      SAVE1, R4
        BTR      IN2+6(R4), CH
        INC      SAVE1
        DEC      COUNT

```

```

TSR      SSAVEG,R1
TSR      (R1)+,R4
RTS      R7
S.38:    SWRITE  TTY,XB
CW:      WAITR   TTY,CW
         SWRITE  TTY,XC
CW:      WAITR   TTY,DW
         SRFAD   KBD,III
WIII:    WAITR   KBD,WIII
         BRN     S.36
III:     WORD    2,0,2,0
SSAVEG:  WORD    0
SAVE1:   WORD    0
XTSUB:   WORD    0
FLAG:    BYTE    0
CH:      BYTE    0
INPUT:   BYTE    1
PRNTR:   BYTE    10
CARDIN:  BYTE    7
PAPIN:   BYTE    5
KYBRD:   BYTE    1
PAPOUT:  BYTE    6
XB:      WORD    2,0,2,3407
XC:      WORD    22,0,22
         BYTE    15,12
         BCI     %INPUT NOT READY,%
IN2:     WORD    80.
ST2:     BYTE    0,0
COUNT:  WORD    0
         . =     .+80.

```

```

;*****
;   THE FOLLOWING ROUTINE READS A 3 DIGIT INPUT & PUTS T IN THE *
;   FOLLOWING WORD AFTER CONVERTING IT TO IT'S BINARY LUE. CALL IS*
;   JMS R7.GET.3 *
;*****

```

```

GET.3:   TSR      R2,-(R1)
         TSR      R3,-(R1)
         TSR      R4,-(R1)
         TSR      #3,R2
         CLR      MQ
GET.31:  JMS      R7,GET
         BCMP     CH,#40
         BR7S     GET.31
         CLR      R3
         BTR      CH,R3
         SUB      #60,R3
         MPY      #10.,R4
         ADD      R3,MQ
         DEC      R2
         BRZC     GET.31
         TSR      MQ,(R7)+
         TSR      (R1)+,R4
         TSR      (R1)+,R3
         TSR      (R1)+,R2
         RTS      R7

```

```

*****
; THE FOLLOWING ROUTINE GENERATES A 10 DIGIT RANDOM NUMBER & PUTS *
; IT IN A BUFFER WHOSE STARTING ADDRESS FOLLOWS THE LL TO THIS *
; *****
; *****
GET10:  TSF      R3,-(R1)
        TSF      R2,-(R1)
        TSF      #10.,R3
        TSF      (R7)+,GEN.1
        JMS      R2,RNDY1
        WORD     0
        CLR      R2
        TSF      R2Y,AD
        DIV      #10.,R2
        BTRK     R2,@GEN.1
        L C      GEN.1
        DEC      R3
        BR7C     GEN.1
        TSF      (R1)+,R2
        TSF      (R1)+,R3
        RTS      R7
GEN.1:  WORD     0
; *****
; THIS ROUTINE READS A 10 DIGIT INPUT & GENERATES IT 4 WORD *
; BINARY EQUIVALENT. *****
; *****
GET.10: TSF      R4,-(R1)
        TSF      R3,-(R1)
        TSF      R2,-(R1)
        TSF      #10.,R2
LPGT10: JMS      R7,GET
        BCOMP    CH,#40
        BR7S     LPGT10
        CLR      R3
        BTRK     CH,R3
        SUB      #60,R3
        BTRK     R3,(R7)+
        DEC      R2
        BR7C     LPGT10
        TSF      (R1)+,R2
        TSF      (R1)+,R3
        TSF      (R1)+,R4
        RTS      R7
; *****
; THIS ROUTINE READS THE AUTHORITY VECTOR SPECIFICAT N. OF ANY *
; MODIFICATION CARD & BUILDS IT'S EQUIVALENT AUTHORI VECTOR *
; *****
AUTHSC: TSF      R4,-(R1)
        TSF      R3,-(R1)
        TSF      R2,-(R1)
        TSF      (R7)+,R4
        TSF      R4,R2
        ADD      #256.,R4
        TSF      #128.,R3
        AU.1:    CLR      (R2)+

```

	DPC	R3
	BRZC	AU.1
	TSR	R4,R3
	SUB	#257.,R3
AU.2:	JMS	R7,GET
	BCMP	CH,#'#
	BRZC	AU.2
AU.2.1:	JMS	R7,AUT.1
AU01:	WORD	0
	CMP	AU01,#1
	BRT	AU.4
	CMP	AU01,#256.
	BRCT	AU.4
	TSR	AU01,R2
	ADD	R3,R2
AU.3:	JMS	R7,AUT.2
AU02:	WORD	0
	RTSR	AU02,(R2)+
	BCMP	R2,R4
	BRZS	AUTEND
	BCMP	AU02+1,#'.
	BRZS	AUTEND
	BCMP	AU02+1,#'##
	BRZC	AU.3
	BRN	AU.2.1
AU.4:	SWRITE	TTY,SME.13
W13:	WAITR	TTY,W13
AUTEND:	TSR	(R1)+,R2
	TSR	(R1)+,R3
	TSR	(R1)+,R4
	RTS	R7
SME.13:	WORD	118.,0,118.
	BYTE	15,12
	BCI	% COMMAND ERROR IN AUTHORITY %
	BCI	% VECTOR MODIFICATION PROGRAM %
	BYTE	15,12
	BCI	% CONTINUES AFTER DELETING PART%
	BCI	% OF COMMAND FOLLOWING ERROR %
AUT.1:	TSR	R2,-(R1)
	CLR	MQ
AUT.12:	JMS	R7,GET
	BCMP	CH,#40
	BRZS	AUT.12
	BCMP	CH,#'.
	BRZS	AUT.13
	MPY	#10.,R2
	RTSR	CH,R2
	SUB	#60,R2
	ADD	R2,MQ
	BRN	AUT.12
AUT.13:	TSR	MQ,(R7)+
	TSR	(R1)+,R2
	RTS	R7
AUT.2:	TSR	R2,-(R1)
	CLR	MQ


```

AUT.21:  JMS      R7,GET
          BCMP     CH,#40
          BRZS     AUT.21
          BCMP     CH,#1.
          BRZS     AUT.22
          BCMP     CH,#1#
          BRZS     AUT.23
          BCMP     CH,#1.
          BRZS     AUT.24
          MPY      #10,R2
          BTSR     CH,R2
          SUB      #60,R2
          ADD      R2,M0
          BRN      AUT.21
AUT.22:  TSR      M0,R2
          BTSR     R2,(R7)+
          BTSR     #1.,(R7)+
          BRN      AUT.23
AUT.23:  TSR      M0,R2
          BTSR     R2,(R7)+
          BTSR     #1#,(R7)+
          BRN      AUT.25
AUT.24:  TSR      M0,R2
          BTSR     R2,(R7)+
          BCLR     (R7)+
AUT.25:  TSH      (R1)+,R2
          RTS      R7
          END

```

SCT.2:	WORD	0
	JMS	R7,ALOCTS
CYL.3:	WORD	0
SCT.3:	WORD	0
	TSR	CYL.3.1SECYL
	TSR	SCT.3.1SECTR
	TSR	#0,R2
	TSR	#18..R6
S.16:	TST	SECSP1(R2)
	HRGE	S.18
S.17:	ADD	#14..R2
	DEC	R6
	BR7C	S.16
	HRN	S.26
S.18:	TSR	R2,R3
	TST	R2
	BR7S	S.19
	TSR	SECSP1(R2),CYL.4
	ADD	#2,R3
	TSR	SECSP1(R3),SCT.4
	JMS	R7,RELSEC
CYL.4:	WORD	0
SCT.4:	WORD	0
	TSR	#-1,SECSP1(R2)
	BRN	S.17
S.19:	TSR	SECSP1(R2),CYL.5
	ADD	#2,R3
	TSR	SECSP1(R3),SCT.5
	JMS	R4,READ
CYL.5:	WORD	0
SCT.5:	WORD	0
	WORD	SECSP2
	TSR	#4,R4
	TSR	#0,R5
S.20:	INC	R5
	CMP	R5,DBSC.1
	BR7S	S.25
	BTSR	#48..SECSP2(R4)
S.25:	ADD	#5,R4
	CMP	R5,#51
	BR7C	S.20
	TSR	CYL.5.CYL.6
	TSR	SCT.5.SCT.6
	JMS	R7,RELSEC
CYL.6:	WORD	0
SCT.6:	WORD	0
	JMS	R7,ALOCTS
CYL.7:	WORD	0
SCT.7:	WORD	0
	TSR	CYL.7.CYL.8
	TSR	SCT.7.SCT.8
	JMS	R4,WRITE
CYL.8:	WORD	0
SCT.8:	WORD	0
	WORD	SECSP2

	TSR	CYL.7,SECSP1(R2)
	TSR	SCT.7,SECSP1(R3)
	HRN	S.17
S.26:	TSR	CYL.3,CYL.9
	TSR	SCT.3,SCT.9
	JMS	R4,WRITE
CYL.9:	WORD	0
SCT.9:	WORD	0
	WORD	SECSP1
ASCINT:	SWRITE	TTY,SMES.9
W9:	WAITR	TTY,W9
	SREAD	KBD,SIN.2
W9:	WAITR	KBD,W9
	BCMP	SIN.2+6,#'Y
	BRZC	ES.30
	TSR	ASECYL,CYL.10
	TSR	ASECTR,SCT.10
	JMS	R4,HEAD
CYL.10:	WORD	0
SCT.10:	WORD	0
	WORD	SECSP1
	TSR	CYL.10,CYL.11
	TSR	SCT.10,SCT.11
	JMS	R7,RELSEC
CYL.11:	WORD	0
SCT.11:	WORD	0
	JMS	R7,ALOCIS
CYL.12:	WORD	0
SCT.12:	WORD	0
	TSR	CYL.12,ASECYL
	TSR	SCT.12,ASECTR
	TSR	#0,R2
	TSR	#18.,TEMI.1
S.27:	TST	SECSP1(R2)
	BRGE	S.29
S.28:	ADD	#14.,R2
	DEC	TEMI.1
	BRZC	S.27
	HRN	S.33
S.29:	TSR	R2,R3
	TSR	SECSP1(R2),CYL.13
	TSR	#-1,SECSP1(R2)
	ADD	#2,R3
	TSR	SECSP1(R3),SCT.13
	JMS	R4,READ
CYL.13:	WORD	0
SCT.13:	WORD	0
	WORD	SECSP2
	TSR	CYL.13,CYL.14
	TSR	SCT.13,SCT.14
	JMS	R7,RELSEC
CYL.14:	WORD	0
SCT.14:	WORD	0
	TSR	#0,R4
	TSR	#25.,TEMI.2

```

S.30:      TST      SECSP2(R4)
          BVC      S.32
S.31:      ADD      #10.,R4
          DFC      TLM1.2
          BRZC     S.30
          BRN      S.28
S.32:      TSR      R4,R5
          TSR      SECSP2(R4),CYL.15
          AND      #2,R5
          TSP      SECSP2(R5),DVO.1
          ADD      #2,R5
          TSR      SECSP2(R5),DVO.2
          JMS      R7,DVORUL
DVO.1:     WORD     0
DVO.2:     WORD     0
DVO.3:     WORD     0
          TSR      DVO.3,SCT.15
          JMS      R7,RELSEC
CYL.12:    WORD     0
SCT.15:    WORD     0
          BRN      S.31
S.33:      TSR      CYL.12,CYL.16
          TSR      SCT.12,SCT.16
          JMS      R4,WRITE
CYL.16:    WORD     0
SCT.16:    WORD     0
          WORD     SECSP1
ES.30:     TSR      SSAVE3,R1
          TSR      (R1)+,R2
          TSR      (R1)+,R3
          TSR      (R1)+,R4
          TSR      (R1)+,R5
          TSR      (R1)+,R6
          JMP      EXIT
SSAVE3:    WORD     0
TEMI.1:    WORD     0
TEMI.2:    WORD     0
SMES.8:    WORD     52.,0,52.
          BYTE     15,12
          BCI      % DO YOU WANT TO INITIALIZE
          BCI      % FIRST SECURITY CODES      %
SMES.9:    WORD     54.,0,54.
          BYTE     15,12
          BCI      % DO YOU WANT TO INITIALIZE
          BCI      % AUTHORITY VECTOR CODES    %
SIN.2:     WORD     2,0,2
          .*=      .+2
          END

```

```

*****
*****
;   MODIFY PROGRAM WHICH ACCEPTS DHA'S REQUEST FOR DELETING, ADDING
;   OR MODIFYING ANY SECURITY CODE OR AUTHORITY VECTOR.
*****
*****

```

```

.=          104100
FLAG=       100706
CARDIN=     100712
KYBRD=      100714
PAPIN=      100713
CH=         100707
SFCSP1=     22170
SFCSP2=     22570
EXIT=       76214
VORBLE=     24400
DVORBI=     24600
READ=       22060
WRITE=      22066
ISFCYL=     26002
ISFCTR=     26004
ASECYL=     26006
ASECTR=     26010
MO=         177736
ALOCYS=     77106
INPUT=      100710
RELSEC=     77550
GET.3=      101104
GET.10=     101262
GIVSEC=     25000
WORBLE=     25400
GFNWOR=     101200
GET=        100400
AUTHSC=     101340

```

```

R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
R6=%6
R7=%7

```

```

MODIFY:      KBD=          0
              TTY=          1
              RESET
W10:         SWRITE      TTY,SME.10
              WAITR      TTY,W10.
              SREAD      KBD,SIN.3
WR6:         WAITR      KBD,WR6      8
              BCLR      FLAG
              CMP        SIN.3+6, #"CA
              BRZC      S.41
              BTR      CARDIN, INPUT
              BRN      S.43
S.41:        CMP        SIN.3+6, #"KY
              BRZC      S.42

```

```

      HTSR      KYBRD,INPUT
      BRN       S.43
S.42:  CMP      SIN,3+6,#"PA
      BR7C      MODIFY
      HTSR      PAPIN,INPUT
S.43:  JMS      R7,GET
      BCMP      CH,#40
      BR7S      S.43
S.44:  BCMP      CH,#'D
      BR7C      S.45
      JMP       S.49
S.45:  BCMP      CH,#'E
      BR7C      S.46
      JMP       S.53
S.46:  BCMP      CH,#'F
      BR7C      S.47
      JMP       EXIT
S.47:  SWRITE   TTY,SME,11
W11:   WAITR    TTY,W11
S.48:  JMS      R7,GET
S.AU:  BCMP      CH,#'
      BR7C      S.48
      BRN       S.43
S.49:  JMS      R7,GET
      BCMP      CH,#40
      BR7S      S.49
      BCMP      CH,#61
      BR7S      S.50
      BCMP      CH,#62
      BR7S      S.51
      BRN       S.47

```

```

;*****
;   THIS SEGMENT DELETES THE FIRST SECURITY CODE AS REQUESTED BY DBA*
;*****

```

```

S.50:  JMS      R7,GET.3
C.1:   WORD     0
      CMP      C.1,#1
      BRLT     S.47
      CMP      C.1,#918.
      BRGT     S.47
      TSR      ISECYL,CYL,21
      TSR      ISECTR,SCT,21
      JMS      R4,READ
CYL.21: WORD     0
SCT.21: WORD     0
      WORD     SE CSP1
      SUB      #1,C.1
      TSR      C,1,MQ
      CLR      C.1
      DIV      #51,.,C.1
      MPY      #14,.,R2
      TSR      MQ,R2
      TSR      SECSP1(R2),CYL,22
      TSR      SECSP1+2(R2),SCT,22
      TST      CYL,22

```

	BRGE	NXTMOD
	JMP	S.52
NXTMOD:	JMS	R4, READ
CYL, 22:	WORD	0
SCT, 22:	WORD	0
	WORD	SECSP2
	TSR	C.1, MQ
	MPY	#5, C.1
	TSR	MQ, R2
	BTSR	#48., SECSP2+4(R2)
	TSR	CYL, 22, CYL, 23
	TSR	SCT, 22, SCT, 23
	JMS	R4, WRITE
CYL, 23:	WORD	0
SCT, 23:	WORD	0
	WORD	SECSP2
	JMP	S.48

THIS SEGMENT DELETES THE AUTHORITY AUGMENTING SECURITY CODE AS		
REQUESTED BY THE DBA.		

S.51:	JMS	R7, GET.3
C.2:	WORD	0
	CMPL	C.2, #1
	BRLT	S.47
	CMPL	C.2, #450.
	BRGT	S.47
	TSR	ASECYL, CYL, 24
	TSR	ASECTR, SCT, 24
	JMS	R4, READ
CYL, 24:	WORD	0
SCT, 24:	WORD	0
	WORD	SECSP1
	SUB	#1, C.2
	TSR	C.2, MQ
	CLR	C.2
	DIV	#25., C.2
	MPY	#14., R2
	TSR	MQ, R2
	TSR	SECSP1(R2), CYL, 25
	TSR	SECSP1+2(R2), SCT, 25
	TSR	CYL, 25
	BRLT	S.52
	JMS	R4, READ
CYL, 25:	WORD	0
SCT, 25:	WORD	0
	WORD	SECSP2
	TSR	C.2, MQ
	MPY	#10., C.2
	TSR	MQ, R2
	TSR	SECSP2(R2)
	BRLT	S.52
	TSR	SECSP2(R2), CYL, 26
	TSR	#-1, SECSP2(R2)
	TSR	SECSP2+2(R2), DVO, 4

```

          TSR          SECSP2+4(R2),DVO.5
          JMS          R7,DVORBL
DVO.4:    WORD        0
DVO.5:    WORD        0
DVO.6:    WORD        0
          TSR          DVO.6,SCT.26
          JMS          R7,RELSEC
CYL.26:   WORD        0
SCT.26:   WORD        0
          TSR          CYL.25,CYL.27
          TSR          SCT.25,SCT.27
          JMS          R4,WRITE
CYL.27:   WORD        0
SCT.27:   WORD        0
          WORD        SECSP2
          JMP          S.48
S.52:     SWRITE      TTY,SME.12
W12:      WAITR       TTY,W12
          JMP          S.48
S.53:     JMS         R7,GET
          BCMP         CH,#40
          BRZS         S.53
          CLR          R6
          BISS        CH,R6
          BCMP        CH,#61
          BRZS        S.57
          BCMP        CH,#62
          BRZC        S.54
          JMP          S.60
S.54:     BCMP        CH,#63
          BRZC        S.55
          JMP          S.60
S.55:     BCMP        CH,#64
          BRZS        S.56
          JMP          S.47
S.56:     JMP         S.66

```

```

*****
1  THIS SEGMENT ENTERS A NEW FIRST SECURITY CODE OR M IFIES AN
1  EXISTING ONE AS REQUESTED BY THE DBA
*****

```

```

S.57:     JMS         R7,GET.3
C.3:      WORD        0
          CMP         C.3,#1
          BRGE        S.E.1
          JMP         S.47
S.E.1:    CMP         C.3,#918
          BRLE        S.E.2
          JMP         S.47
S.E.2:    TSR         ISECYL,CYL.28
          TSR         ISECTR,SCT.28
          JMS         R4,READ
CYL.28:   WORD        0
SCT.28:   WORD        0
          WORD        SECSP1
          SUB         #1,C.3

```


	TSR	C.3,M0
	CLR	C.3
	DIV	#51.,C.3
	MPY	#14.,R2
	TSR	M0,R2
	TST	SECSP1(R2)
	BRLT	S.E.5
	TSR	SECSP1(R2),S.E.3
	TSR	SECSP1+2(R2),S.E.4
	JMS	R4,READ
S.E.3:	WORD	0
S.E.4:	WORD	0
	WORD	SECSP2
	BRN	S.59
S.E.5:	JMS	R7,ALOCIS
CYL.29:	WORD	0
SCT.29:	WORD	0
	TSR	CYL.29,SECSP1(R2)
	TSR	SCT.29,SECSP1+2(R2)
	TSR	#4,R3
	TSR	#51.,R4
S.58:	BTSR	#48.,SECSP2(R3)
	ADD	#5,R3
	DEC	R4
	BR7C	S.58
	TSR	#SECSP1+4,WOR.1
	ADD	R2,WOR.1
	JMS	R7,GENWOR
WOR.1:	WORD	0
S.59:	TSR	#SECSP1+4,WOR.2
	ADD	R2,WOR.2
	JMS	R7,GET.10
SE.1:	.=	.,+10.
	TSR	#SE.1,WOR.3
	TSR	#SE.1,WOR.4
	JMS	R7,WORBLE
WOR.2:	WORD	0
WOR.3:	WORD	0
	JMS	R7,GIVSEC
WOR.4:	WORD	0
WOR.5:	.=	.,+4
	TSR	C.3,M0
	MPY	#5,C.3
	TSR	M0,R3
	BCIR	SECSP2+4(R3)
	BTSR	WOR.5,SECSP2(R3)
	BTSR	WOR.5+1,SECSP2+1(R3)
	BTSR	WOR.5+2,SECSP2+2(R3)
	BTSR	WOR.5+3,SECSP2+3(R3)
	TSR	SECSP1(R2),CYL.30
	TSR	SECSP1+2(R2),SCT.30
	JMS	R4,WRITE
CYL.30:	WORD	0
SCT.30:	WORD	0

```

        WORD      SECSP2
        TSR       ISECYL,CYL,31
        TSR       ISECTR,SCT,31
        JMS       R4,WRITE
CYL.31:  WORD      0
SCT.31:  WORD      0
        WORD      SECSP1
        JMP       S.48
;*****
;   THIS SEGMENT ENTERS A NEW AUTHORITY AUGMENTING SECURITY CODE OR *
;   MODIFIES AN EXISTING ONE WITH IT'S ASSOCIATED VECTOR KEPT INTACT*
;*****
S.60:    JMS      R7,GET.3
C.4:     WORD      0
        CMP      C.4,#1
        BRGE     S.E.6
        JMP      S.47
S.E.6:   CMP      C.4,#450.
        BRLE     S.E.7
        JMP      S.47
S.E.7:   TSR      ASECYL,CYL,32
        TSR      ASECTR,SCT,32
        JMS      R4,READ
CYL.32:  WORD      0
SCT.32:  WORD      0
        WORD      SECSP1
        SUB      #1,C.4
        TSR      C.4,MQ
        CLR      C.4
        DIV      #25.,C.4
        MPY      #14.,R2
        TSR      MQ,R2
        TST      SECSP1(R2)
        BRLT     S.E.10
        TSR      SECSP1(R2),S.E.8
        TSR      SECSP1+2(R2),S.E.9
        JMS      R4,READ
S.E.8:   WORD      0
S.E.9:   WORD      0
        WORD      SECSP2
        BRN      S.62
S.E.10:  JMS      R7,ALOCTS
CYL.33:  WORD      0
SCT.33:  WORD      0
        TSR      CYL.33,SECSP1(R2)
        TSR      SCT.33,SECSP1+2(R2)
        CLR      R3
        TSR      #25.,R4
S.61:    TSR      #-1,SECSP2(R3)
        ADD      #10.,R3
        DEC      R4
        BRZC     S.61
        TSR      #SECSP1+4,WOR.6
        ADD      R2,WOR.6
        JMS      R7,GENWOR

```

```

WOR.6:  WORD      0
J.62:   TSR      #SECSP1+4,WOR.7
        ADD      R2,WOR.7
        JMS      R7,GET.10
GET.2:   L=      .+10.
        TSR      #SE.2,WOR.8
        TSR      #SE.2,WOR.9
        JMS      R7,WORBLE
WOR.7:   WORD      0
WOR.8:   WORD      0
        JMS      R7,GIVSEC
WOR.9:   WORD      0
WOR.10:  L=      .+4
        TSR      C.4,MQ
        MPY      #10.,C.4
        TSR      MQ,R3
        RTSR     WOR.10,SECSP2+6(R3)
        RTSR     WOR.10+1,SECSP2+7(R3)
        RTSR     WOR.10+2,SECSP2+8(R3)
        RTSR     WOR.10+3,SECSP2+9(R3)
        TSR      SECSP1(R2),CYL.36
        TSR      SECSP1+2(R2),SCT.36
        TSR      ASECYL,CYL.34
        TSR      ASECTR,SCT.34
        JMS      R4,WRITE
CYL.34:  WORD      0
SCT.34:  WORD      0
        WORD      SECSP1
        TST      SECSP2(R3)
        BRLT     S.E.13
        TSR      CYL.36,S.E.11
        TSR      SCT.36,S.E.12
        JMS      R4,WRITE
S.E.11:  WORD      0
S.E.12:  WORD      0
        WORD      SECSP2
        BRN      S.64
S.E.13:  JMS      R7,ALOCIS
CYL.35:  WORD      0
SCT.35:  WORD      0
        TSR      CYL.35,SECSP2(R3)
        TSR      SCT.35,VO.1
        JMS      R7,VORBLE
VO.1:    WORD      0
VO.2:    WORD      0
VO.3:    WORD      0
        TSR      VO.2,SECSP2+2(R3)
        TSR      VO.3,SECSP2+4(R3)
        TSR      VO.3,SECSP2+4(R3)
        JMS      R4,WRITE
CYL.36:  WORD      0
SCT.36:  WORD      0
        WORD      SECSP2
        CLR      R2
S.63:    CLR      SECSP1(R2)

```

```

      INC      R2
      INC      R2
      CMP      R2,#256
      BRZC     S.63
      BCMP     R6,#62
      BRZC     S.65
      TSN      CYL.35,CYL.37
      TSN      SCT.35,SCT.37
      JMS      R4,WRITE
CYL.37: WORD    0
SCT.37: WORD    0
      WORD     SECSPI
      JMP      S.48
S.64:  BCMP     R6,#62
      BRZC     S.E.14
      JMP      S.48
S.E.14: TSN      SECSPI2(R3),CYL.35
      TSN      SECSPI2+2(R3),DVO.7
      TSN      SECSPI2+4(R3),DVO.8
      JMS      R7,DVORBL
DVO.7: WORD    0
DVO.8: WORD    0
DVO.9: WORD    0
      TSN      DVO.9,SCT.35
S.62:  JMS      R7,AUTHSC
      WORD     SECSPI
      TSN      CYL.35,CYL.38
      TSN      SCT.35,SCT.38
      JMS      R4,WRITE
CYL.38: WORD    0
SCT.38: WORD    0
      WORD     SECSPI
      JMP      S.AU

```

```

*****
:   THIS SEGMENT CHANGES THE AUTHORITY VECTOR ASSOCIATED WITH A
:   GIVEN SECURITY CODE WHILE LEAVING THE CODE INTACT
*****

```

```

S.66:  JMS      R7,GET.3
C.5:   WORD     0
      CMP      C.5,#1
      BRGE     S.E.15
      JMP      S.47
S.E.15: CMP      C.5,#450
      BRLE     S.E.16
      JMP      S.47
S.E.16: TSN      ASECYL,CYL.39
      TSN      ASECIR,SCT.39
      JMS      R4,READ
CYL.39: WORD    0
SCT.39: WORD    0
      WORD     SECSPI
      SUB      #1,C.5
      TSN      C.5,MQ
      CLN      C.5
      DIV      #25,C.5

```

```

      MPY      #14.,R2
      TSN      MQ,R2
      TST      SECSP1(R2)
      RRGU     S.67
      JMP      G.47
S.67:  TSN      SECSP1(R2),CYL.40
      TSN      SECSP1+2(R2),SCT.40
      JMS      R4,READ
CYL.40: WORD    0
SCT.40: WORD    0
      WORD     SECSP2
      TSN      C.5,MQ
      MPY      #10.,C.5
      TSN      MQ,R3
      TST      SECSP2(R3)
      RRGU     S.E.14
      JMP      S.47
SME.10: WORD    66.,0,66.
      BYTE     15,12
      HCI      % SPECIFY THE DEVICE CONTAINING%
      HCI      % SECURITY CODES MODIFICATION CARDS%
SME.11: WORD    44.,0,44.
      BYTE     15,12
      HCI      % ILLEGAL MODIFICATION INSTRUCTION %
      HCI      %OMITTED.%
SME.12: WORD    60.,0,60.
      BYTE     15,12
      HCI      % DELETING AN EMPTY CODE.%
      HCI      %NO ACTION TAKEN.PROGRAM CONTINUES.%
SIN.3: WORD    8.,0,8.
      .+8.
      END

```

```

;*****
;*****
;  REMOVE PROGRAM WHICH REMOVES A RELATION FROM THE DATA BASE AS
;  REQUESTED BY THE DBA
;*****
;*****

```

```

      L=          66400
      R1=         %1
      R2=         %2
      R3=         %3
      R4=         %4
      R5=         %5
      R6=         %6
      R7=         %7
      TTY=        1
      KBD=        0
      MD=         177736
      FLAG=       100706
      INPUT=      100710
      NIND=       30626
      RELID=      30366
      KEYRP=      31016
      FMKRP=      27600
      FLNGT=      46156
      FLRID=      35516
      FLUNM=      31516
      FOMAT=      36502
      FDIST=      37466
      FDLISC=     60006
      PMINX=      30506
      PRINXC=     50472
      PRINDX=     40060
      BITMAP=     27740

```

```

      BITMAP=     27740
      GET.3=     101104
      PUT=       66000
      EXIT=      76214

```

```

REMOVE:  BCLR      FLAG
         BTSR      #1,INPUT
         SWRITE    TTY,RME.3
WHM2:    WAITR     TTY,WHM2
         JMS       R7,GET.3

```

```

R.1:     WORD      0
         CMP       R.1,#39.
         BRGT      R.021
         TST       R.1
         BRLE      R.021
         TSR       R.1,R2
         CLR       R3
         BTSR      NIND(R2),R3
         LSL       R2
         TST       RELID(R2)
         BRZS      R.021
         CLR       RELID(R2)
         CLR       KEYRP(R2)
         TSR       FMKRP(R2),R.01

```

	JMS	R5,FLNGT
	WORD	R.01
	WORD	R.02
	ADD	#3,R.02
	TSO	R.02,MU
	MPY	R3,R.01
	TSR	M0,R.01
	TSR	#372,R6
	CLR	R3
R.2:	BCMP	R.1,FLDID(R3)
	BR7S	R.3
	ADD	#2,R3
	DEC	R6
	BR7C	R.2
R.021:	RESET	
	SWRITE	TTY,RME.1
RW1:	WAITR	TTY,RW1
	JMP	EXIT
R.3:	TSR	R3,R5
	TSR	#1,R4
	ADD	#2,R3
R.4:	BCMP	R.1,FLDID(R3)
	BR7C	R.5
	INC	R4
	ADD	#2,R3
	BRN	R.4
R.5:	TSR	FLDID(R3),FLDID(R5)
	TSR	FOMAT(R3),FOMAT(R5)
	LSL	R3
	LSL	R3
	LSL	R5
	LSI	R5
	TSR	FLDNM(R3),FLDNM(R5)
	TSR	FLDNM+2(R3),FLDNM+2(R5)
	TSR	FLDNM+4(R3),FLDNM+4(R5)
	TSR	FLDNM+6(R3),FLDNM+6(R5)
	ASR	R3
	ASR	R3
	ASR	R3
	ASR	R5
	ASR	R5
	ASR	R5
	RTSR	FDIST(R3),FDIST(R5)
	INC	R3
	INC	R5
	CMP	R3,FDLISC
	BRGE	R.6
	LSL	R3
	LSL	R5
	BRN	R.5
R.6:	CMP	R5,FDLISC
	BRGE	R.61
	LSL	R5
	CLR	FLDID(R5)
	ASR	R5

```

      INC      R5
      BRN      R.6
R.61:  SUB      R4,FDLISC
      TSH      PMINX(R2),R3
      ASR      R2
      ADD      R.02,R3
      SUB      #3,R3
      CLR      R4
      HTSR     NIND(R2),R4
      LSL      R2
R.071: HTSR     PRINDX(R3),R.072
      JMS      R7,RELCYL
R.072: WORD     0
      ADD      R.02,R3
      DEC      R4
      BRGT     R.071
      TSH      PMINX(R2),R3
      TSH      R3,R6
      SUB      R.01,PRINXC
      TSH      R.01,R5
      ADD      R3,R5
R.7:   HTSR     (R5)+,(R3)+
      CMP      R3,PRINXC
      BRGE     R.8
      BRN      R.7
R.8:   TSH      #2,R2
R.9:   TST      RELID(R2)
      BRZS     R.10
      CMP      R6,PMINX(R2)
      BRGT     R.10
      SUB      R.01,PMINX(R2)
R.10:  ADD      #2,R2
      CMP      R2,#80
      BRLT     R.9
      JMP      PUT
R.01:  WORD     0
R.02:  WORD     0
RME.1: WORD     52.,0,52.
      BYTE     15,12
      BCI      % DELETING A NON EXISTING%
      BCI      % RELATION,COMMAND IGNORED %
RME.3: WORD     66.,0,66.
      BYTE     15,12
      BCI      % SPECIFY THE RELATION NO. OF%
      BCI      % RELATION TO BE REMOVED %
      BCI      %IN 3 DIGITS %

```

```

*****
;   THIS ROUTINE IS USED TO RELEASE THE CYLINDERS VACATED BY THE
;   RELATION WHICH IS REMOVED FROM THE DATA BASE
*****

```

```

RELCYL: TSH      R2,-(R1)
        TSH      R3,-(R1)
        TSH      R4,-(R1)
        TSH      (R7)+,R2
        TST      R2

```



```

R.15:  BRIT      R.20
      TSH      #24.,R3
      CMP      R2,#16.
      BRIT     R.17
      SUB      #16.,R2
      SUB      #2,R3
      TST      R3
      BRLT     R.20
      BRN      R.15
R.16:  WORD     0
R.17:  TSH      #1,R.16
R.18:  TST      R2
      BRZS     R.19
      DEC      R2
      LSI      R.16
      BRN      R.18
R.19:  CIB      R.16,BITMAP(R3)
      TSH      (R1)+,R4
      TSH      (R1)+,R3
      TSH      (R1)+,R2
      RTS      R7
R.20:  SWRITE   TTY,RME.2
RW2:   WAITR    TTY,RW2
      STOP
RME.2:  WORD    40.,0,40.
      BYTE     15,12
      HCI      % CYLINDER TO BE RELEASED OUT%
      HCI      % OF RANGE %
      END

```

```

*****
*****
; PUT ROUTINE WHICH PUTS RELATION DIRECTORY, FIELD LIST & PRIMARY
; INDEX TABLE & OTHER VARIABLE DATA FROM CORE INTO DISK.
*****
*****
      = 66000
      R1= %1
      R2= %2
      R3= %3
      R4= %4
      R5= %5
      R6= %6
      R7= %7
      ND= 26146
      ND1= 26150
      WRITE= 22066
      EXIT= 76214
PUT:   TSR      ND, R2
      JMS      R4, WRITE
      WORD     201..101..50400
      JMS      R4, WRITE
      WORD     201..138..60000
      TSR      ND1, R3
      TSR      #25..R5
P1:    TSR      (R3)+, R4
      TSR      (R3)+, STR10
      TSR      (R3)+, CYL10
P2:    TSR      (R3)+, SCT10
      JMS      R4, WRITE
CYL10: WORD     0
SCT10: WORD     0
STR10: WORD     0
      ADD      #400, STR10
      DEC      R5
      BR7C     P3
      JMP      EXIT
P3:    DEC      R4
      BRGT     P2
      DEC      R2
      BRGT     P1
      JMP      EXIT
      END

```